

COURSE TEXTBOOK

Algorithmic Game Theory

Games, equilibria, mechanisms and computation

Sercan Külcü

Giresun University · Department of Computer Engineering

Algorithmic Game Theory

Sercan Külcü
Giresun University, Department of Computer Engineering
Giresun, Türkiye

This book is the printed form of the course notes published at
`sercankulcu.github.io/teaching/algorithmic-game-theory/`

Slides, study questions, the games and the working versions of the interactive figures are on that page. The interactive figures in this book are printed with their default values. The code for each chapter is in the `github.com/sercankulcu/algorithmic-game-theory-python-en` repository.

© Sercan Külcü. All rights reserved.

Generated from the notes of 24 November 2026.

Preface

Algorithmic game theory studies strategic interaction through a computer scientist's eyes: it combines the models of game theory with algorithms and complexity analysis. This book is made up of the notes for the Algorithmic Game Theory course taught in the Department of Computer Engineering at Giresun University, and its chapters follow the order of the course.

The book has five parts. In order, they are **Tools and Foundations** (Chapters 0–3), **Mechanisms** (Chapters 4–7), **Networks and Efficiency** (Chapters 8–9), **Cooperation and Society** (Chapters 10–12), and **Dynamics and the Future** (Chapters 13–14). Chapter 0 rebuilds the mathematics the course uses (probability, expected value, optimization) through game examples; readers who know these topics can start directly with Chapter 1.

Every chapter follows the same layout. It opens with learning outcomes. Formal definitions, theorems and worked examples are set in named boxes. *Common mistake* boxes point out the misconceptions students fall into most often. Code traces show, step by step, how the calculations are done with short Python functions. Each chapter closes with a summary, a list of key terms, *Test Yourself* questions with their answers, suggestions for further reading and a bibliography. The glossary at the end of the book defines each key term in a sentence and names the chapter that explains it.

The code for each chapter is in its own folder in the companion repository `github.com/sercankulcu/algorithmic-game-theory-python-en`. The interactive figures in the notes are printed in this book with their default values. Slides, study questions and the games played in class are on the course page.

Sercan Külcü
Giresun

Contents

Course Map 1

PART I · TOOLS AND FOUNDATIONS

0 Preliminaries 4

0.1 Introduction 5
0.2 Sets and functions 6
0.3 Sums and maxima 12
0.4 Exponents, logarithms, and differentiation rules 22
0.5 Probability and expected value 29
0.6 Mixed strategies 44
0.7 The payoff matrix 49
0.8 Best response 53
0.9 Equations and indifference 57
0.10 Game trees and growth rates 63
0.11 Common mistakes 68
0.12 Summary 69
0.13 Key terms 70
0.14 Test Yourself 71
0.15 Further reading 73
0.16 References 73

1 Decision Theory 74

1.1 Expected utility 74
1.2 Risk attitudes 85
1.3 Bayesian decision 96
1.4 Decision trees and the value of information 105
1.5 Discussion: better decisions under risk and uncertainty 112
1.6 Summary 114
1.7 Key terms 115
1.8 Test Yourself 115
1.9 Further reading 118
1.10 References 118

2 Games and Representations 119

2.1 Core concepts 120
2.2 Representing games 132
2.3 Classifying games 141
2.4 Equilibrium and the Prisoner's Dilemma 145
2.5 Classic 2×2 games 151
2.6 The guess-two-thirds game 156
2.7 Combinatorial games 162

2.8 Sequential games and backward induction 173
2.9 Minimax and alpha-beta pruning 184
2.10 The algorithmic view and mechanism design 212
2.11 Solved problems 215
2.12 Summary 217
2.13 Key terms 218
2.14 Test Yourself 219
2.15 Further reading 222
2.16 References 223

3 Nash Equilibrium 225

3.1 Foundations 225
3.2 Best response 231
3.3 Elimination of dominated strategies 245
3.4 Mixed strategies 254
3.5 Existence and equilibrium selection 264
3.6 Minimax and complexity 272
3.7 Discussion and checklist 282
3.8 Summary 283
3.9 Key terms 285
3.10 Test Yourself 285
3.11 Further reading 287
3.12 References 288

PART II · MECHANISMS

4 Mechanism Design 291

4.1 Basic concepts 291
4.2 Direct mechanisms 296
4.3 The VCG mechanism 313
4.4 Mechanisms without money 321
4.5 Bayesian games 340
4.6 Review 345
4.7 Summary 348
4.8 Key terms 349
4.9 Test Yourself 349
4.10 Further reading 352
4.11 References 352

5 Auctions 354

5.1 Basic concepts 354
5.2 The four basic formats 361
5.3 Sealed-bid strategies 366
5.4 Revenue and information 376
5.5 Two-sided markets 389
5.6 Review 396

5.7 Summary 398
5.8 Key terms 399
5.9 Test Yourself 400
5.10 Further reading 402
5.11 References 402

6 Online Platforms 404

6.1 Platforms and markets 405
6.2 Network effects and critical mass 409
6.3 Platform pricing 418
6.4 Sponsored search 427
6.5 Reputation and competition 444
6.6 Discussion 451
6.7 Summary 452
6.8 Key terms 453
6.9 Test Yourself 453
6.10 Further reading 456
6.11 References 457

7 Social Choice and Voting 458

7.1 Social choice 459
7.2 Voting rules 463
7.3 The Condorcet paradox 478
7.4 Strategic voting 488
7.5 Impossibility theorems 491
7.6 Computational social choice and applications 499
7.7 Summary 504
7.8 Key terms 504
7.9 Test Yourself 505
7.10 Further reading 507
7.11 References 508

PART III · NETWORKS AND EFFICIENCY

8 Network Design and Routing 510

8.1 Networks and routing 510
8.2 Selfish routing 517
8.3 Braess's paradox 529
8.4 Congestion games 538
8.5 Network formation 545
8.6 Control and incentives 556
8.7 Summary 560
8.8 Key terms 561
8.9 Test Yourself 562
8.10 Further reading 564
8.11 References 564

9 The Price of Anarchy 566

9.1 Basic concepts 566
9.2 Pigou's network 574
9.3 Linear latency and Braess 580
9.4 Load balancing 592
9.5 The Price of Stability 602
9.6 Reducing the PoA 608
9.7 Worked problems 613
9.8 Summary 615
9.9 Key terms 617
9.10 Test Yourself 617
9.11 Further reading 619
9.12 References 619

PART IV · COOPERATION AND SOCIETY

10 Multi-Agent Systems 622

10.1 Agents and multi-agent systems 622
10.2 Cooperation and coordination 629
10.3 Task allocation 637
10.4 Coalitional games 649
10.5 The public goods game 670
10.6 Summary 676
10.7 Key terms 677
10.8 Test Yourself 678
10.9 Further reading 680
10.10 References 681

11 Algorithmic Fairness and Ethics 682

11.1 Introduction 682
11.2 Fairness criteria 685
11.3 Incompatibility of the criteria 700
11.4 Fair division 709
11.5 Ethics and governance 726
11.6 Summary 734
11.7 Key terms 735
11.8 Test Yourself 736
11.9 Further reading 738
11.10 References 738

12 Behavioral and Collective Decisions 740

12.1 Behavioral decision making 741
12.2 Social preferences 755
12.3 Collective decisions and common resources 761
12.4 Public goods and punishment 768
12.5 Threshold projects 776

12.6 Collective information 779
12.7 Common mistakes in collective decisions and example problems 788
12.8 Summary 790
12.9 Key terms 791
12.10 Test Yourself 791
12.11 Further reading 794
12.12 References 794

PART V · DYNAMICS AND THE FUTURE

13 Learning and Adaptation 797
13.1 Introduction 797
13.2 Best response and fictitious play 801
13.3 Regret and Hedge 810
13.4 The multi-armed bandit 822
13.5 Reinforcement learning 831
13.6 Evolutionary dynamics 835
13.7 Repeated games 840
13.8 Discussion: learning, humans, and ethics 848
13.9 Summary 849
13.10 Key terms 850
13.11 Test Yourself 850
13.12 Further reading 853
13.13 References 853

14 Future Directions 855
14.1 Introduction 856
14.2 AI agents 859
14.3 Learning mechanisms 873
14.4 Blockchain mechanisms 884
14.5 Scale and open questions 894
14.6 Review: common mistakes and worked problems 899
14.7 Summary 902
14.8 Key terms 903
14.9 Test Yourself 903
14.10 Further reading 905
14.11 References 906

APPENDICES

A Glossary of Terms 907

B Prerequisites for Algorithmic Game Theory 928

C Important People in Algorithmic Game Theory 949

ALGORITHMIC GAME THEORY

Course Map

This page puts every chapter of the course on one map. Each chapter is a node, and an arrow shows which idea from an earlier topic the next one relies on. Read it before the course starts to see where the topics are heading, or during the course to see where you are.

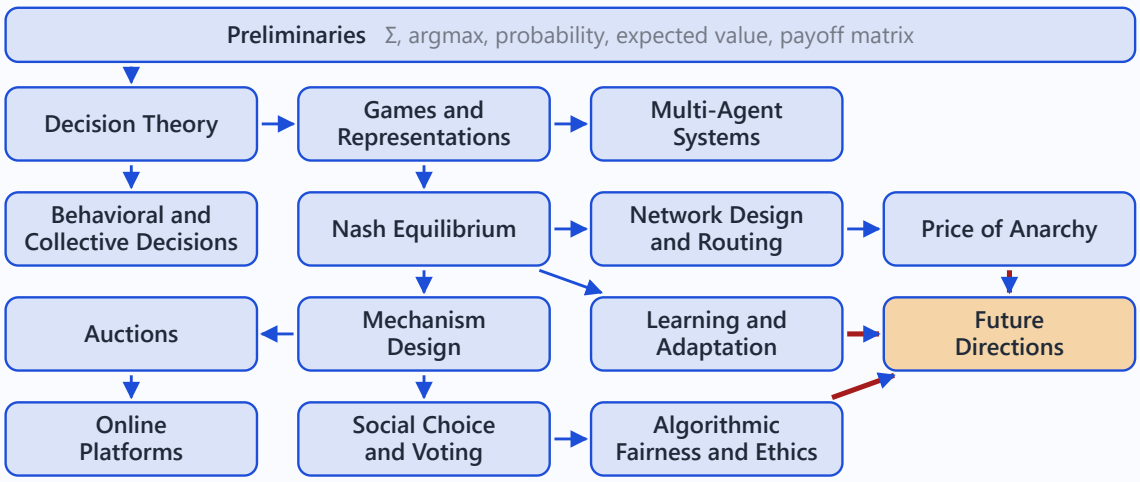


Figure 1: The course map. Each arrow points from a topic to one that builds on it. Every path starts at Preliminaries, and the main branches meet in Future Directions.

Figure 1 shows the connections between the topics. Each node is a chapter, and an arrow indicates which idea of the earlier topic the later one relies on.

Topic (chapter)	Builds on	Connection
Preliminaries (0)	—	Σ and argmax, probability and expected value, reading a payoff matrix, the indifference equation; every chapter uses these tools.
Decision Theory (1)	Preliminaries	The single-person decision: expected utility, risk attitudes, Bayesian decisions and the value of information.
Games and Representations (2)	Decision Theory	When decision makers multiply, a game arises: normal and extensive form, backward induction.
Nash Equilibrium (3)	Games and Representations	The equilibria marked in 2×2 games lead to the general Nash definition and the existence theorem.
Mechanism Design (4)	Nash Equilibrium	Equilibrium + incentives: instead of solving a game, design the rules that produce the desired equilibrium.

Topic (chapter)	Builds on	Connection
Auctions (5)	Mechanism Design	The single-item case of VCG is the second-price (Vickrey) auction.
Online Platforms (6)	Auctions	Ad-slot auctions: GSP is the multi-slot generalization of Vickrey's idea.
Social Choice and Voting (7)	Mechanism Design	Social choice function → voting rules; Gibbard–Satterthwaite.
Network Design and Routing (8)	Nash Equilibrium	Congestion games: the Wardrop equilibrium in nonatomic routing (Beckmann potential), the pure Nash equilibrium in finite atomic games (Rosenthal potential).
Price of Anarchy (9)	Network Design and Routing	Equilibrium quality in Pigou and Braess networks → a price of anarchy of 4/3; congestion tolls.
Multi-Agent Systems (10)	Games and Representations	The cooperative / noncooperative distinction → coalitional games, the Shapley value.
Algorithmic Fairness and Ethics (11)	Social Choice and Voting	From collective decisions to fairness: criteria and fair division.
Behavioral and Collective Decisions (12)	Decision Theory	People deviate from expected utility (loss aversion); with common goods, individual and collective interest diverge.
Learning and Adaptation (13)	Nash Equilibrium	How is an equilibrium found? Players can converge to equilibrium by learning.
Future Directions (14)	Price of Anarchy, Learning and Adaptation, Algorithmic Fairness and Ethics	Scale, learning agents and ethics come together in today's open questions.

Table 1: The connections on the map.

The backbone of the map can be read as **equilibrium → design → application → learning → future**. We move from the single-person decision to games, and from games to equilibrium. The equilibrium concept then opens in two directions: designing the rules (mechanisms, auctions, voting) and measuring the outcome of existing rules (networks, the price of anarchy). Learning asks how equilibrium is reached. The topics of the last chapter (Future Directions) sit where these branches meet. For example, algorithmic collusion rests on both repeated games and learning, and transaction fee mechanisms rest on both auctions and congestion pricing.

PART I

Tools and Foundations

Chapters 0–3

PART I · CHAPTER 0

Preliminaries

Game theory studies the interdependent choices of several decision makers. To state those choices precisely and to compute their consequences, a small but solid body of mathematics is needed: sets and functions, sums and maxima, derivatives, probability and expected value, reading matrices, simple equations, and trees. These notes introduce the tools used throughout the course one at a time, from scratch, and always through game examples. Each concept is first explained in plain language, then defined formally, and followed by a fully worked example with every step shown and a common mistake. The notes are written so that a reader with a weak mathematical background can use them on their own.

LEARNING OUTCOMES

Notation
Read sets, Cartesian products, and function notation such as $u_1(s_1, s_2)$.

Σ, max, argmax
Evaluate sums and geometric series; find the maximum value and the choice that attains it, using derivatives (chain rule, partial derivatives) when the choice is continuous.

Probability
Compute probabilities by counting, by multiplying, and by conditioning (Bayes); work with $U[0, 1]$.

Expected value
Compute the expected payoff of a lottery and of a mixed strategy; read expressions with exponentials, ln, and softmax.

Matrices and best responses
Read an (a, b) pair correctly and mark best responses in a payoff matrix.

Equations and trees
Solve quadratic and indifference equations; solve a game tree from the leaves to the root; read growth rates with O and Θ.

0.1 Introduction

This chapter introduces no new game-theoretic concept. Its purpose is to present the language that game theory speaks. That language consists of a few symbols, a few rules of calculation, and a few habits of reading, and all of it is made concrete through one small game: Rock-Paper-Scissors.

0.1.1 Starting with a game: Rock-Paper-Scissors

Two players simultaneously choose Rock, Paper, or Scissors. Rock beats Scissors, Scissors beats Paper, and Paper beats Rock; equal choices end in a tie. The winner gains 1 point, the loser gives up 1 point, and in a tie nobody scores. Figure 0.1 shows this beating relation as a cycle: each arrow points from a move to the move it beats.

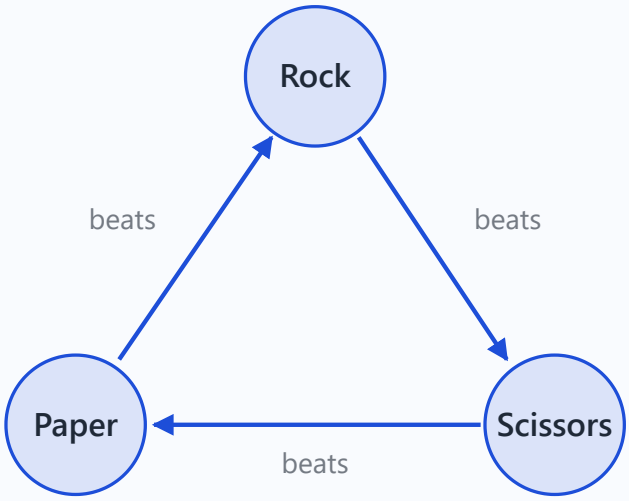


Figure 0.1: The beating cycle of Rock-Paper-Scissors. Rock beats Scissors, Scissors beats Paper, Paper beats Rock. No move beats all the others.

The cycle has an important consequence: there is no single "best move." Rock is good against Scissors but bad against Paper. Which move is good depends entirely on what the opponent plays. That dependence is the whole subject of game theory: a player's payoff depends not only on her own choice but also on the choice of the opponent.

Suppose we play against a computer. The computer can behave in two ways. In the first, it picks a random move each round. In the second, it watches our moves: it learns how often we play each move, or which move we tend to play after which, and plays accordingly. Against the second kind of computer, two questions arise naturally:

- Is there a way of playing that a learning computer cannot beat in the long run?

- If someone always plays Rock and the opponent notices, how many points per round does she lose on average?

The tools needed to answer these questions are the following: set and function notation to write down moves and outcomes; probability and expected value for the question "how many points on average"; argmax and best response for the question "which move is best"; and the indifference equation for "a way of playing that cannot be beaten." Once these tools have been built up in turn, both questions are answered completely (0.9.5).

0.1.2 The role of mathematics in game theory

The first role of mathematics in game theory is as a *notation*. The sentence "If Ayşe plays Rock and Bora plays Scissors, Ayşe gets 1 point" is long and ambiguous; the same information is written briefly and exactly as $u_1(Rock, Scissors) = 1$. As games grow, this brevity becomes a necessity: it is impossible to describe all the outcomes of a ten-player game in sentences.

The second role is *computation*. When a player asks "if my opponent plays Rock half of the time, what should I do?", the answer is a comparison of numbers: compute the average payoff of each move and pick the largest. Probability, sums, and finding a maximum are enough for this. When the choice is continuous (such as how many units a firm produces), the maximum is found with a derivative.

The third role is *measurement*. How long a method will take on a computer is described by its growth rate as a function of the input size. This is where the "algorithmic" in algorithmic game theory comes from: is it possible to find an equilibrium, and if so, how fast? The language of these questions is O and Θ notation (0.10.3).

Every section of these notes follows the same order: first a plain explanation of the concept, then its formal definition, then why it matters, then a worked example, and finally the most common mistake. The numbers in the examples are the same as the numbers in the slides.

0.2 Sets and functions

Writing down a game takes three things: who plays, what options each player has, and who gets what in each situation. The options are written as a set, "each situation" as a Cartesian product, and "who gets what" as a function. The counting rules are closely tied to these three structures as well.

0.2.1 Sets and the Cartesian product

A set is an unordered collection of objects. In game theory the most common set is the list of a player's options. A set is written with curly braces: *{Rock, Paper, Scissors}*. Order does not matter in a set; *{Rock, Paper}* and *{Paper, Rock}* are the same set. An element is not written twice in a set.

Definition 0.1

If a is an element of a set A, we write $a \in A$, read "a is an element of A." The number of elements of a set is written $|A|$.

The **Cartesian product** of two sets A and B is the set of all ordered pairs whose first element comes from A and whose second element comes from B:

$$A \times B = \{(a, b) : a \in A, b \in B\}$$

The sizes multiply: $|A \times B| = |A| \cdot |B|$.

The word "ordered" in the definition is crucial. Order does not matter in a set, but it does in a pair: in the pair (a, b), the first slot belongs to A and the second to B. Two pairs are equal only if the elements in both slots are the same.

In game theory, a player's options are called **strategies**, and the set of all of a player's options is called her **strategy set**. Player 1's strategy set is written S_1 and player 2's S_2. In one round of the game each player picks a strategy. The pair formed by the two players' choices together is called a **strategy profile**. The set of all possible profiles is exactly the Cartesian product $S_1 \times S_2$. The Cartesian product is thus a complete answer to the question "what can happen in the game?"

Example 0.1

Ayşe and Bora play Rock-Paper-Scissors. Ayşe's strategy set is $S_1 = \{Rock, Paper, Scissors\}$.

1. Rock is one of Ayşe's strategies: $Rock \in S_1$. The set has three elements: $|S_1| = 3$.
2. Bora's options are the same: $S_2 = \{Rock, Paper, Scissors\}$, $|S_2| = 3$.
3. A profile is built by writing Ayşe's choice in the first slot and Bora's in the second. Each of Ayşe's choices is paired with each of Bora's: three pairs with Rock, three with Paper, three with Scissors.
4. The total number of profiles is $|S_1 \times S_2| = 3 \cdot 3 = 9$.

The nine profiles are arranged as a three-by-three table in Figure 0.2. Rows are Ayşe's choices, columns are Bora's; each cell is a profile.

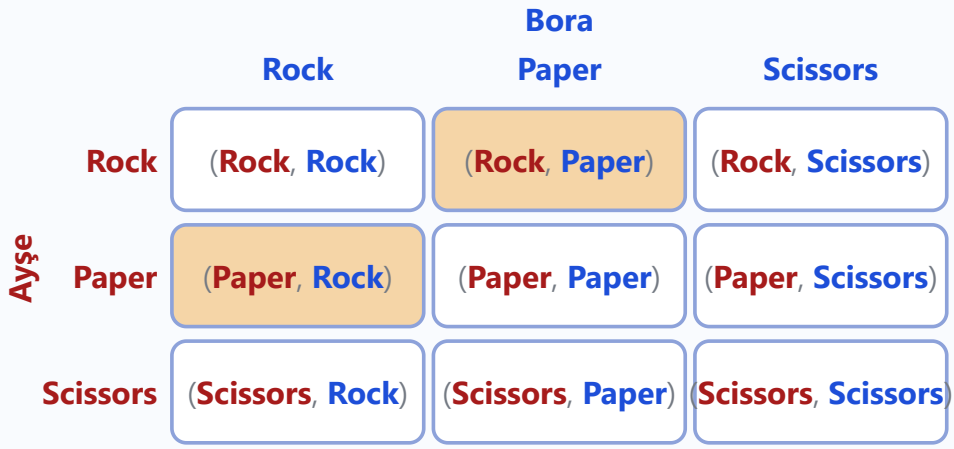


Figure 0.2: $S_1 \times S_2$: the nine strategy profiles of Rock-Paper-Scissors. Rows are Ayşe's choices (red), columns are Bora's (blue). The two orange cells (Rock, Paper) and (Paper, Rock) are different profiles.

The table form is no accident. The profiles of every two-player game can be arranged in such a table; when payoffs are written in the cells, the table becomes a **payoff matrix** (0.7). In a three-player game, profiles are lists of three elements: $(s_1, s_2, s_3) \in S_1 \times S_2 \times S_3$. Such lists are called triples, or n-tuples in general. The counting rule is the same: the sizes multiply.

Common mistake

Treating (Rock, Paper) and (Paper, Rock) as the same. Order matters in a pair: the first element is Ayşe's choice and the second is Bora's. In the profile (Rock, Paper), Ayşe played Rock and Bora played Paper, and Bora wins. In (Paper, Rock) the roles are swapped and Ayşe wins. These are two different outcomes, one won by Ayşe and the other by Bora (the orange cells in Figure 0.2). This confusion also leads to a wrong count: someone who thinks without order finds six outcomes instead of nine.

0.2.2 Function notation

A function can be pictured as a machine: it takes an input and returns *exactly one* output. The same input always gives the same output. The most important function in game theory is the **payoff function**. Its input is a strategy profile and its output is a number: the points a player receives when that profile is played.

Definition 0.2

The notation $f : X \to Y$ says that f is a function that assigns to each element of the set X exactly one element of the set Y. The element assigned to x is written $f(x)$.

The payoff function of player i is $u_i : S_1 \times S_2 \to \mathbb{R}$, where $\mathbb{R}$ is the set of real numbers. The notation reads "from profiles to real numbers."

The subscript in the notation says whose payoff is being looked at. For the same profile, u_1 gives Ayşe's payoff and u_2 gives Bora's. These two numbers are usually different. The two strategies inside the parentheses carry the profile, that is, the answer to "who played what?"; their order is the same as in the Cartesian product.

Example 0.2

Let us read a few payoff values in Rock-Paper-Scissors one by one.

1. $u_1(\text{Rock, Scissors})$: Ayşe played Rock and Bora played Scissors. Rock beats Scissors, so Ayşe gets 1 point. The value is 1.
2. $u_2(\text{Rock, Scissors})$: The same profile, but now we ask for Bora's payoff. Bora lost and gives up 1 point. The value is -1.
3. $u_1(\text{Paper, Paper})$: A tie. The value is 0.
4. $u_1(\text{Rock, Paper})$: Paper beats Rock; Ayşe is the loser. The value is -1.

In this game $u_1 + u_2 = 0$ in every profile: what one player wins, the other loses. Games with this property are called **zero-sum games**.

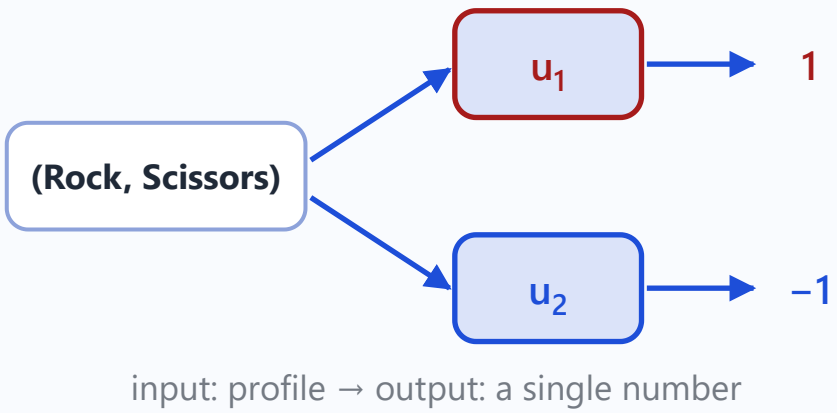


Figure 0.3: A payoff function is like a machine: the input is a profile (Rock, Scissors) and the output is a single number. The machine u_1 gives 1 and the machine u_2 gives −1.

Figure 0.3 shows the same input going into two different machines. In a game each player has her own payoff function. Describing a game completely means giving, for every player, the value of this function at every profile. In two-player Rock-Paper-Scissors that comes to $2 \cdot 9 = 18$ numbers.

Why is function notation so important? Because every definition in game theory is written in it. For example, the sentence "If Ayşe knows that Bora plays Scissors, she prefers Rock to Paper" is expressed by the inequality $u_1(\text{Rock, Scissors}) > u_1(\text{Paper, Scissors})$, that is, $1 > -1$. Once the notation can be read, the definitions can be read too.

Common mistake

Reading the value of $u_1(\text{Rock, Paper})$ as "Paper wins." The value of the function is a number, not a word: $u_1(\text{Rock, Paper}) = -1$. The subscript says whose payoff is being looked at; u_1 and u_2 give different numbers for the same profile. Someone who skips the subscript may mistake Bora's payoff for Ayşe's.

0.2.3 Counting: the product rule, n!, and C(n, k)

A frequent question in game theory is "how many?": how many strategy profiles are there, how many pure strategies does a player have, how many leaves does a tree have? Three simple counting rules answer these questions. The same rules are used in probability calculations, because for equally likely outcomes a probability is a ratio of counts (0.5.1).

Definition 0.3

- **Product rule**: In successive choices, if the first choice has a options and the second has b, the two choices can be made together in $a \cdot b$ different ways. With more choices, the numbers of options are multiplied in the same way.
- **Factorial**: n distinct objects can be arranged in $n! = n \cdot (n-1) \cdot \ldots \cdot 2 \cdot 1$ different orders. By definition $0! = 1$.
- **Combination (binomial coefficient)**: the number of ways to choose k of n distinct objects when order does not matter is

$$C(n, k) = n! / (k! \cdot (n-k)!)$$

The product rule is the general form of the rule for the size of a Cartesian product. The factorial is a special application of the product rule: the first position of an ordering is filled by one of n objects, the second by one of the remaining $n - 1$, and so on. Combinations count choices in which order is irrelevant. The formula can be understood as follows: choosing k of n objects *in order* can be done in n

$\cdot (n-1) \cdot \ldots \cdot (n-k+1) = n!/(n-k)!$ ways. In this count the same k objects have been counted separately in $k!$ different orders. When order is irrelevant, each group must be counted once, so we divide by $k!$.

Example 0.3

1. **Number of profiles.** Three players have 2, 3, and 4 strategies respectively. A profile is built by choosing one strategy from each player. By the product rule: $2 \cdot 3 \cdot 4 = 24$ strategy profiles.
2. **Orderings.** Three people are to be lined up. The first position can be filled by 3 people, the second by the remaining 2, the third by the remaining 1: $3! = 3 \cdot 2 \cdot 1 = 6$ different orders.
3. **The guard's choice.** A guard can protect 2 of 4 targets. Which two targets she protects is one of her pure strategies. Order is irrelevant; protecting A first and then B is the same strategy as protecting B first and then A. The count is $C(4, 2) = 4!/(2! \cdot 2!) = 24/4 = 6$: {A, B}, {A, C}, {A, D}, {B, C}, {B, D}, {C, D} (Figure 0.4).
4. **A larger example.** If 5 of 20 targets are to be protected, there are $C(20, 5) = (20 \cdot 19 \cdot 18 \cdot 17 \cdot 16)/(5 \cdot 4 \cdot 3 \cdot 2 \cdot 1) = 1{,}860{,}480 / 120 = 15{,}504$ pure strategies.

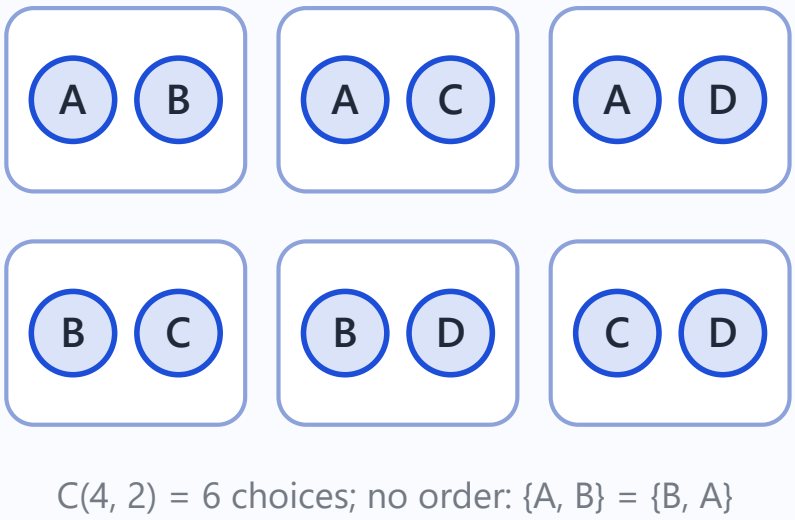


Figure 0.4: The $C(4, 2) = 6$ ways to choose 2 of 4 targets. There is no order: {A, B} and {B, A} are the same choice.

The growth in the last example is striking. When the number of targets goes from 4 to 20 and the number of protected targets from 2 to 5, the number of pure strategies goes from 6 to 15,504. In real security applications (airports, ports, forest patrols) the number of targets is in the hundreds. Methods that list strategies one by one therefore quickly become inadequate. This rapid growth in the number of strategies gives rise to one of the basic questions of algorithmic game theory: among an astronomical number of strategies, can a computer find the best one quickly?

Common mistake

Counting order in a choice where order is irrelevant. Choosing from 4 targets as "first one, then another" gives $4 \cdot 3 = 12$ ordered pairs. But {A, B} and {B, A} are the same protection; every choice has been counted twice. The correct number is $12 / 2! = 6$. The opposite mistake also occurs: in a Cartesian product order matters, and (Rock, Paper) and (Paper, Rock) must be counted separately. A single question decides which rule to use: "does swapping the order give something different?"

0.3 Sums and maxima

Most computations in game theory consist of two operations: adding things up and picking the largest. An expected payoff is a sum; a best response is a choice of the largest value; the total payoff in a repeated game is an infinite sum. When there are infinitely many options, the maximum is found with a derivative.

0.3.1 The summation symbol Σ

Σ is the capital Greek letter "sigma" and means "add them all up." It lets long sums be written briefly. The expression below the symbol says where the summation starts, the number above it says where it stops, and the expression to its right is what is added at each step.

Definition 0.4

$$\Sigma_{i=1}^{n} x_i = x_1 + x_2 + \ldots + x_n$$

Here i is a *counter*: first it is 1, then 2, ..., and finally n. For each value of the counter we write x_i and add them all.

The name of the counter does not matter; writing j or k instead of i does not change the sum. What matters is which values the counter runs through and what is added for each value. In some sums the limits are not written and the summation runs "over all possible values." For example, $\Sigma_x \, x \cdot P(X = x)$ adds one term for every value x that X can take.

Two rules are used often when working with Σ. First, a constant factor common to every term can be taken outside the sum: $\Sigma \, c \cdot x_i = c \cdot \Sigma \, x_i$. Second, the sum of two sums can be written as a single sum: $\Sigma \, (x_i + y_i) = \Sigma \, x_i + \Sigma \, y_i$. Both rules follow from the fact that the order and grouping of addition do not change the result.

Example 0.4

A die is rolled three times and shows 4, 1, and 6 in turn: $x_1 = 4, x_2 = 1, x_3 = 6$ (Figure 0.5).

1. Write the sum out: $\Sigma_{i=1}^{3} x_i = x_1 + x_2 + x_3$.
2. Substitute the values: $4 + 1 + 6 = 11$.
3. On a fair die each face has probability $p_i = 1/6$. The probabilities of the six faces add up to $\Sigma_{i=1}^{6} p_i = 6 \cdot 1/6 = 1$. This uses the constant-factor rule: all six terms are the same. The fact that probabilities sum to 1 holds for every distribution.
4. An expected value is also a sum: $\Sigma_{i=1}^{n} x_i \cdot p_i$. This sum is treated in detail in 0.5.5.

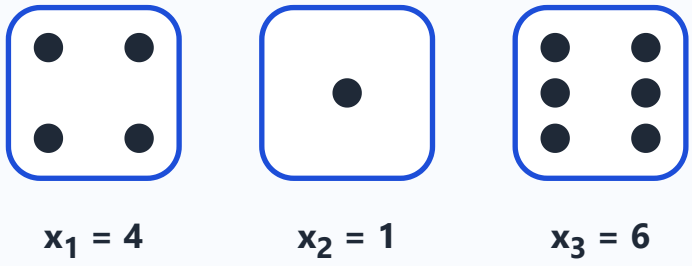


Figure 0.5: Three die rolls: $x_1 = 4, x_2 = 1, x_3 = 6$. Their sum is $\Sigma_{i=1}^{3} x_i = 11$.

The Σ notation is the mathematical counterpart of a loop in programming. $\Sigma_{i=1}^{3} x_i$ means "total = 0; for each i from 1 to 3, add x[i] to total." This similarity is plain in the code that computes the expected value of a lottery (0.5.6).

Common mistake

Moving the square outside the sum. $(\Sigma x_i)^2$ and Σx_i^2 are different. The first adds first and then squares: $(4 + 1 + 6)^2 = 11^2 = 121$. The second squares each term first and then adds: $4^2 + 1^2 + 6^2 = 16 + 1 + 36 = 53$. The right procedure is to evaluate the expression inside Σ separately for each i and then add the results. A constant factor can be taken outside, but a square or any other operation cannot.

0.3.2 Geometric series and the discount factor δ

Some games are played once and end; others are played again and again between the same players. Two firms set prices every month; two neighbors meet every day. In repeated games, today's payoff has to be compared with future payoffs. People generally prefer a payoff today to the same payoff later; moreover, it may be uncertain whether the game will continue into the next round. Both effects are modeled with a single number, the **discount factor**.

Definition 0.5

The discount factor is a number $0 \le \delta < 1$. A payoff received t rounds from now is multiplied by δ^t when viewed from today: a payoff of 1 today is worth 1, one round later it is worth δ, two rounds later δ^2.

If the payoff in every round is 1, the infinite sum starting today is a **geometric series**:

$$1 + \delta + \delta^2 + \delta^3 + \ldots = 1/(1 - \delta)$$

At first sight this formula is surprising: infinitely many positive numbers are added, yet the result is a finite number. The reason is that the terms shrink quickly. For $\delta = 1/2$ the terms are 1, 1/2, 1/4, 1/8, ..., halving at every step (Figure 0.6). The sum gets a little closer to 2 at each step but never exceeds it.

Theorem 0.1 (Geometric series)

If $0 \le \delta < 1$, then $S = \Sigma_{t=0}^{\infty} \delta^t = 1/(1 - \delta)$.

Proof. Denote the sum by S: $S = 1 + \delta + \delta^2 + \delta^3 + \ldots$. Multiplying both sides by δ raises the exponent of every term by one: $\delta S = \delta + \delta^2 + \delta^3 + \ldots$. Writing the two lines one under the other, we see that every term of the second line also appears in the first; the only extra term in the first line is the leading 1. So subtracting the second line from the first leaves only that term: $S - \delta S = 1$. The left side is written $S(1 - \delta)$. Since $\delta < 1$, the number $1 - \delta$ is nonzero and both sides can be divided by it: $S = 1/(1 - \delta)$. (This derivation assumes that S is a finite number; for $\delta < 1$ the assumption is correct, because the terms shrink quickly.) □

If the payoff in each round is not 1 but a, the constant-factor rule makes the sum $a/(1 - \delta)$. If the sum starts one round from now rather than today, each term carries an extra δ: $\delta + \delta^2 + \ldots = \delta \cdot (1 + \delta + \ldots) = \delta/(1 - \delta)$.

Example 0.5

1. **δ = 0.9 and a payoff of 3 each round.** The sum is $3 + 3 \cdot 0.9 + 3 \cdot 0.9^2 + \ldots = 3/(1 - 0.9) = 3/0.1 = 30$.
2. **$\delta = 1/2$ and a payoff of 1 each round.** $1 + 1/2 + 1/4 + \ldots = 1/(1 - 1/2) = 1/(1/2) = 2$. The first few partial sums are 1, 1.5, 1.75, 1.875, approaching 2.
3. **A sum that starts tomorrow.** If, with $\delta = 0.9$, the payoff of 3 per round starts tomorrow rather than today, the sum is $3 \cdot 0.9/(1 - 0.9) = 2.7/0.1 = 27$. The difference is today's 3: $30 - 27 = 3$.

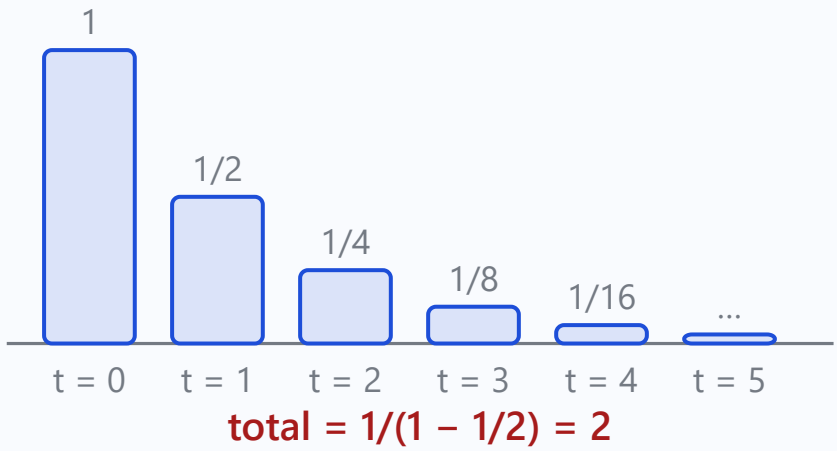


Figure 0.6: Present values of the rounds for $\delta = 1/2$: 1, 1/2, 1/4, 1/8, 1/16, ... The bars halve at every step; the total is $1/(1 - 1/2) = 2$.

The importance of this formula in game theory is as follows. In a repeated game, if two players cooperate continually, each receives a certain payoff every round; if one cheats once, she gains more in that round but may be punished in later rounds. Whether cheating pays is decided by comparing the one-time gain with the present value of the payoffs that will be lost later. That present value is exactly a geometric series. When δ is large (the players are patient, or the game is likely to continue), the future weighs heavily and cooperation can be sustained.

Common mistake

Forgetting the first term. A sum that begins "after today" is $\delta + \delta^2 + \ldots = \delta/(1 - \delta)$, not $1/(1 - \delta)$. Which formula to use is chosen by looking at the round in which the first payoff arrives. A second mistake is to use the formula for $\delta = 1$: then the denominator is zero and the formula is invalid. Indeed, the sum $1 + 1 + 1 + \ldots$ grows without bound.

0.3.3 max and argmax

When a player decides, she looks at what each of her options would bring and picks the one that brings the most. This thought contains two separate questions: "how much can be won at most?" and "which option brings the most?" The answer to the first is a number; the answer to the second is an option. The two questions are written with two different notations.

Definition 0.6

Let A be the set of options and let $f(a)$ be the value of each option.

- $\max_{a \in A} f(a)$: the largest of the values. It is a *number*.
- $\operatorname{argmax}_{a \in A} f(a)$: the *set* of options that attain the largest value. If there is no tie it has one element, and that one element is often written instead.

In game theory a player chooses an action, not a value. That is why argmax is used more often. A player's "best response" is the argmax of her own payoff against the opponent's choice (0.8). The max then tells the payoff that this best response brings.

Example 0.6

One of four face-down cards is to be chosen. The prizes on the cards are A: 3, B: 7, C: 2, D: 5 chips.

1. Write the values side by side: $f(A) = 3, f(B) = 7, f(C) = 2, f(D) = 5$.
2. Find the largest value: $\max\{3, 7, 2, 5\} = 7$. This is a number.
3. Find the option that gives this value: only B gives 7. $\operatorname{argmax} = B$. This is an option and it is the decision itself: "choose B."

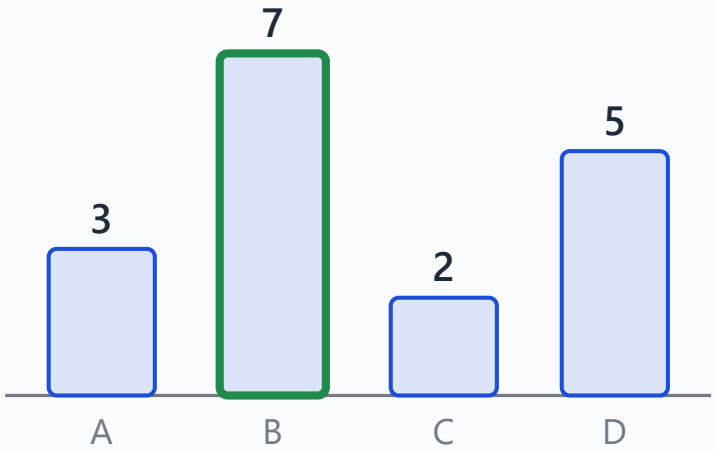


Figure 0.7: The prizes of the four cards. The height of the tallest bar is max = 7, and its label is argmax = B (green frame).

Figure 0.7 separates the two concepts visually: max is the *height* of the bar, argmax is the *label under* the bar. The distinction may look unimportant at first, but in many definitions of game theory max and argmax appear together on the same line. Writing one in place of the other makes the definition meaningless.

Common mistake

Writing the value in place of the argmax ("argmax = 7") or the reverse ("max = B"). The second trap is a tie. If the prizes were 3, 7, 2, 7, the argmax would not be a single element but the set *{B, D}*; both are best choices. Picking one "at random" in a tie and forgetting the other causes some equilibria to be overlooked in game theory, because a player's indifference between two options is the basis of mixed strategies (0.9.3).

0.3.4 Code trace: finding the argmax with a loop

A computer finds an argmax with a loop. The Python code below is the same as the code in the slides. This time the prizes are deliberately chosen with a tie: 3, 7, 2, 7. B and D are tied for the largest, so we can see how the code behaves on a tie.

```
action = ["A", "B", "C", "D"]
payoff = [3, 7, 2, 7]
best = 0               # index of the best
for i in range(1, 4):
    if payoff[i] > payoff[best]:
        best = i
print(payoff[best], action[best])
```

The lines of the code work as follows:

- `action` holds the names of the options and `payoff` holds the prizes in the same order. The two lists are matched by index: index 0 corresponds to A and to 3, index 1 to B and to 7.
- `best = 0`: at the start the first card (A, 3 chips) is taken to be the best. The variable holds an *index*, not a *value*; this is the code counterpart of the fact that an argmax is an option.
- The loop runs for `i = 1, 2, 3`. `range(1, 4)` starts at 1 and does not include 4. Index 0 is skipped because the first card is already the starting candidate.
- At each step the prize of the card being examined is compared with the prize of the best so far. If it is strictly larger, the best is updated.
- At the end the prize of the best (the max) and its name (the argmax) are printed.

Step	i	Comparison	Result	best
Start	–	–	A is the candidate	0 (A, 3)
1	1	*7 > 3*?	yes	1 (B, 7)
2	2	*2 > 7*?	no	1 (B, 7)
3	3	*7 > 7*?	no	1 (B, 7)
Output	–	–	`7 B`	–

Table 0.1: Trace of the argmax loop (prizes 3, 7, 2, 7).

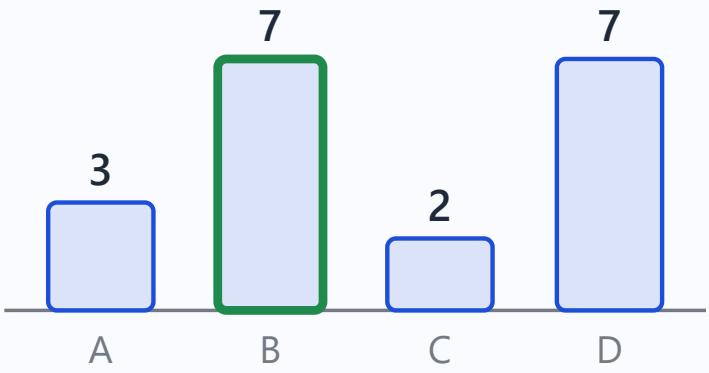


Figure 0.8: The final state of the trace. The green frame marks B, the best card so far. Although D also gives 7, the code returns only B; the true argmax set is {B, D}.

Table 0.1 shows every step of the loop and Figure 0.8 shows the final state. The third step of the trace table contains an important detail. At the third step D's prize is 7 and the prize of the best is also 7. Because the comparison is strict "greater than" (`>`), *7 > 7* is false and the best does not change. On a tie the code keeps the *first one found*. The output is `7 B`. Mathematically, however, the argmax set is *{B, D}*. The code has returned only one element of the set.

If the comparison were `>=`, the last tied element would be kept and the output would be `7 D`. To find all the best options, two passes are made: first the max is found, then all indices whose prize equals the max are collected in a list. This detail matters in game-theory code. For example, a program that finds all equilibria of a matrix will miss some equilibria if it skips tied maxima.

The amount of work of the code is also easy to count. For n options the loop runs $n - 1$ times and makes one comparison per round. The work grows in proportion to the number of options; in the notation of 0.10.3 this is $\Theta(n)$. Since the maximum cannot be found without looking at every option at least once, there is no faster method.

0.3.5 Maximum of a function: derivative equals zero

When the options are limited to a few cards, an argmax is read off a table. But some choices are continuous. A firm can produce 4 units, or 4.5 units, or 4.37 units. Because the number of options is infinite, no table can be written. In this case the maximum is found using the shape of the function's graph.

Think of the top of a hill. Climbing up, the path slopes upward; going down, it slopes downward. Right at the summit the path is flat for a moment: the slope is zero. The same holds for a smooth payoff curve: to the left of the peak the slope is positive, to the right it is negative, and at the peak it is zero. The function that gives the slope of a function at every point is called the **derivative** and is written $f'(q)$.

Definition 0.7

$f'(q)$ is the slope of the graph of f at the point q: it says how fast f changes when q increases by a tiny amount. The rules needed in these notes are:

- $(q^k)' = k \cdot q^{k-1}$; for example $(q^2)' = 2q$ and $(q)' = 1$.
- The derivative of a constant is 0.
- $(c \cdot f)' = c \cdot f'$: a constant factor comes out of the derivative.
- The derivative of a sum is the sum of the derivatives: $(f + g)' = f' + g'$.

The derivative of the derivative is called the **second derivative** and is written f''. If $f'' < 0$, the slope is decreasing; a point where the slope is zero is then a peak.

The method has three steps: take the derivative, set it equal to zero, and solve the resulting equation. The point found is called a candidate, because not every point where the slope is zero is a peak. The slope is also zero at the bottom of a valley. A fourth step is therefore needed: check that the candidate really is a peak.

Example 0.7

We look for the maximum of the function $f(q) = q(10 - q)$.

1. **Expand.** Multiply out: $f(q) = 10q - q^2$.
2. **Derivative.** The derivative of $10q$ is 10 and the derivative of q^2 is $2q$. By the sum rule, $f'(q) = 10 - 2q$.
3. **Set to zero.** $10 - 2q = 0 \Rightarrow 2q = 10 \Rightarrow q = 5$.
4. **Value.** $f(5) = 5 \cdot (10 - 5) = 25$.
5. **Check.** The second derivative is $f''(q) = -2$, which is negative; the slope is decreasing, so the point is a peak. Neighboring values confirm it: $f(4) = 4 \cdot 6 = 24$ and $f(6) = 6 \cdot 4 = 24$; both are smaller than 25.

Result: $\text{argmax} = 5$, $\max = 25$.

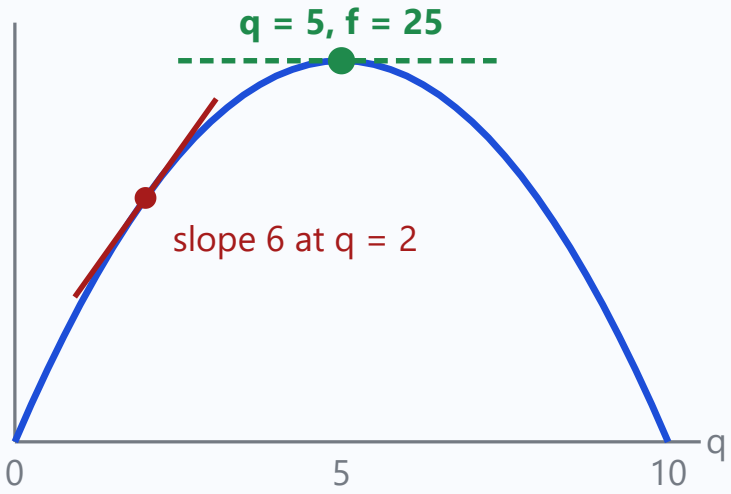


Figure 0.9: *f(q) = q(10 − q)*. At the peak (*q = 5, f = 25*) the tangent is horizontal and the slope is zero. At *q = 2* the slope is *f′(2) = 10 − 4 = 6* (red tangent).

Figure 0.9 shows the geometric meaning of the derivative. The red line at $q = 2$ is tangent to the curve at that point and has slope 6: when q increases by a small amount, f increases by about 6 times as much. At the peak the tangent is horizontal. To the right of the peak, for example at $q = 8$, the slope is $10 - 16 = -6$: as q grows, f falls.

Common mistake

Never checking the point where $f' = 0$. It may be a valley; for example, for $g(q) = q^2$ we have $g'(0) = 0$, but 0 is the bottom of a valley. That it is a peak is shown by $f'' < 0$ or by neighboring values ($f(4) = f(6) = 24 < 25$). If the choice is restricted to an interval, the endpoints must also be examined: sometimes the maximum is not where the slope is zero but at an end of the allowed interval.

0.3.6 The best quantity via derivatives

The first major application of the derivative method in game theory is to firms choosing production quantities. Two firms sell the same good. The market price depends on total production: the more is produced, the lower the price. A firm's profit depends not only on its own quantity but also on the opponent's quantity. This is a game.

The numbers of the model are as follows: the price is $12 - (q + r)$, where q is the firm's quantity and r is the opponent's. There is no production cost. Profit is quantity times price:

$$u(q) = q \cdot (12 - q - r)$$

Example 0.8

The opponent produces $r = 4$ units. We look for the firm's best quantity.

1. **Profit expression.** $u(q) = q \cdot (12 - q - 4) = q(8 - q) = 8q - q^2$.
2. **Derivative.** $u'(q) = 8 - 2q$.
3. **Set to zero.** $8 - 2q = 0 \Rightarrow q = 4$.
4. **Profit.** $u(4) = 4 \cdot (8 - 4) = 16$. The price is $12 - 8 = 4$; 4 units are sold at 4 each. The profit curve and its peak are shown in Figure 0.10.
5. **Check.** $u'' = -2 < 0$. Neighbors: $u(3) = 3 \cdot 5 = 15$ and $u(5) = 5 \cdot 3 = 15$; both are smaller than 16 ✓.

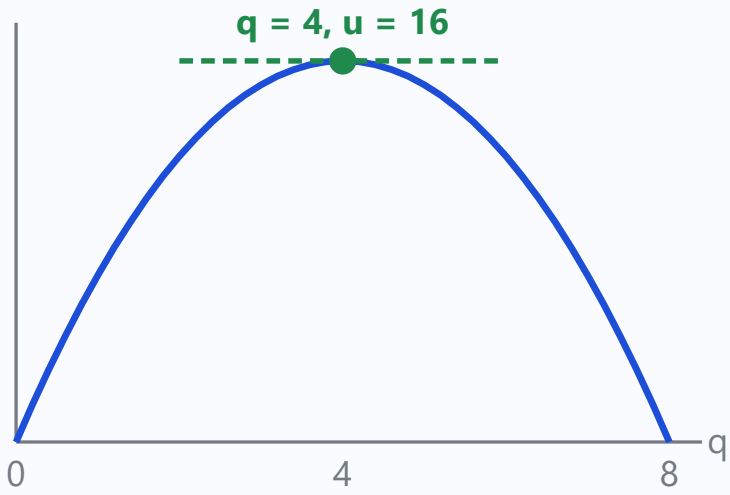


Figure 0.10: The firm's profit $u(q) = q(8 - q)$ when the opponent produces $r = 4$. The peak is at $q = 4$, with profit 16; at $q = 8$ the price falls to zero.

The same computation could be repeated for every quantity of the opponent. Instead, we hold r like a constant and do the computation once, in general. Expanding the profit gives $u(q) = 12q - q^2 - qr$. Since r is constant, the derivative of qr with respect to q is r (constant-factor rule). The derivative is $u'(q) = 12 - 2q - r$, and setting it to zero gives

$$12 - 2q - r = 0 \Rightarrow q = (12 - r)/2$$

Putting in $r = 4$ gives $q = 8/2 = 4$, the result of the example. For $r = 2$, $q = 5$; for $r = 6$, $q = 3$: the more the opponent produces, the less the firm produces.

The formula has a limit: the quantity cannot be negative. If $r \geq 12$ the formula gives zero or a negative number. In that case the price is already zero or below, and the firm's best choice is not to produce at all. The full formula is:

$$q^*(r) = \max\{0, (12 - r)/2\}$$

This formula, which gives the best answer to every quantity of the opponent, is the continuous version of the **best response** that is marked in a matrix in 0.8. In a matrix the best response is found by taking an argmax separately for each column; here the derivative gives the argmax for all the infinitely many "columns" in a single formula. The quantities at which the two firms best-respond to each other simultaneously are found with partial derivatives in 0.4.4.

Common mistake

Thinking that the opponent's quantity changes together with one's own. When the firm chooses q it cannot change r; for the firm r is a constant. So the derivative of the term qr is r, not $q + r$ or $2qr$. The second mistake is to forget the limit: for $r = 14$ the formula gives $q = -1$, but negative production is meaningless and the correct answer is $q^* = 0$.

0.4 Exponents, logarithms, and differentiation rules

In game theory, exponential expressions show up in three places: discounting in repeated games (δ^t), the utility of a risk-averse person ($x^{0.88}$ or $\ln x$), and probabilistic choice rules ($e^{\lambda v}$). Working with these expressions requires the rules for exponents and logarithms, and finding their maxima requires two new differentiation rules: the product rule and the chain rule. For functions of several variables, partial derivatives are used.

0.4.1 Exponent rules and fractional exponents

In the expression x^a, x is called the *base* and a the *exponent*. If a is a positive integer, x^a is x multiplied by itself a times: $x^3 = x \cdot x \cdot x$. The exponent rules follow directly from this definition and are then extended in the same way to fractional and negative exponents.

Definition 0.8

For $x > 0$ and any real numbers a, b:

- $x^a \cdot x^b = x^{a+b}$ (exponents add when multiplying with the same base),
- $(x^a)^b = x^{a \cdot b}$ (exponents multiply for a power of a power),
- $x^0 = 1$,
- $x^{-a} = 1/x^a$,
- $x^{1/2} = \sqrt{x}$.

The reason for the rules is seen by counting. $x^2 \cdot x^3$ is the product of two x's and three x's; five x's are multiplied in all, that is, x^5. The rule $x^0 = 1$ also comes from the first rule: we must have $x^a \cdot x^0 = x^{a+0} = x^a$, so x^0 must be the number that leaves a product unchanged, namely 1. The negative exponent follows by the same logic: since $x^a \cdot x^{-a} = x^0 = 1$, x^{-a} is the reciprocal of x^a. Finally, $x^{1/2} \cdot x^{1/2} = x^1 = x$; the positive number whose square is x is $\sqrt{x}$.

Fractional exponents (such as $x^{0.88}$) obey the same rules. Their meaning can be stated in one sentence: if the exponent is positive and less than 1, x^a increases ever more slowly as x grows. If the exponent is greater than 1, it increases ever faster. Figure 0.11 shows these two cases side by side, together with the case in between, exponent 1.

Example 0.9

1. $\delta^2 \cdot \delta^3 = \delta^{2+3} = \delta^5$. The discount for two rounds ahead combined with the discount for three further rounds is the discount for five rounds.
2. $2^{-1} = 1/2$; $9^{1/2} = \sqrt{9} = 3$; $5^0 = 1$.
3. **Utility.** Let the utility a person gets from x lira of money be measured by $x^{0.88}$. A calculator gives $100^{0.88} \approx 57.5$ and $200^{0.88} \approx 105.9$.
4. When the money doubles, by what factor does the utility grow? By the exponent rule, $(2x)^{0.88} = 2^{0.88} \cdot x^{0.88}$. The ratio is $2^{0.88} \approx 1.84$: $105.9 / 57.5 \approx 1.84$. The utility does not double; it grows by a factor of about 1.84.

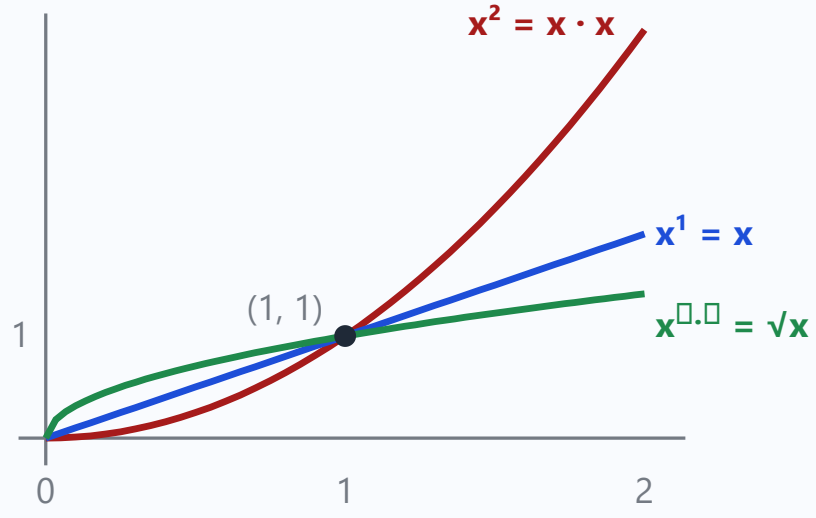


Figure 0.11: x^2 (red), x^1 (blue), and $x^{0.5} = \sqrt{x}$ (green). All three pass through (1, 1). If the exponent is less than 1 the curve flattens as it rises; if it is greater, it steepens.

The meaning of the last example is important in decision theory. A utility function with an exponent below 1 expresses in numbers the idea that "the second hundred lira does not please as much as the first hundred." Such a person, offered a sure 100 lira or an even chance of 0 or 200 lira, chooses the

sure one: the sure option gives utility 57.5, while the lottery gives on average $(0 + 105.9)/2 \approx 53$. This behavior is called risk aversion.

Common mistake

Adding exponents in a sum as well: $2^2 + 2^3 = 4 + 8 = 12$, but $2^5 = 32$. Exponents add only *when multiplying with the same base*; there is no exponent rule for sums. A second common mistake is to think $x^0 = 0$. In fact $x^0 = 1$ (for $x \neq 0$); that is why the discount on today's payoff is $\delta^0 = 1$ and today's payoff counts in full.

0.4.2 The number e and the natural logarithm ln

Some bases are more useful than others. The most important is the number $e \approx 2.718$. What makes e special has to do with the derivative: the slope of the function e^x at every point equals its own value. This property simplifies computations greatly. In game theory, e appears in probabilistic choice rules (0.6.3) and in learning algorithms.

Definition 0.9

e^x is positive for every x. The **natural logarithm** $\ln$ is the inverse operation of e^x: $\ln(e^x) = x$, and for $y > 0$, $e^{\ln y} = y$. That is, $\ln y$ answers the question "to what power must e be raised to give y?" The rules are:

- $\ln(a \cdot b) = \ln a + \ln b$,
- $\ln(a^k) = k \cdot \ln a$,
- $\ln 1 = 0$.

Derivatives: $(e^x)' = e^x$ and $(\ln x)' = 1/x$.

The logarithm rules are the exponent rules read backwards. If we write $a = e^m$ and $b = e^n$, then $a \cdot b = e^{m+n}$; taking the ln of both sides gives $\ln(a \cdot b) = m + n = \ln a + \ln b$. The logarithm turns a product into a sum. This property also explains why algorithms that work with products of many probabilities use logarithms: instead of multiplying small numbers, working with the sum of their logarithms is both easier and numerically safe.

Example 0.10

1. **ln 8.** $8 = 2^3$. By the rule $\ln(a^k) = k \cdot \ln a$, $\ln 8 = 3 \cdot \ln 2 \approx 3 \cdot 0.693 = 2.079$. Round-trip check: $e^{2.079} \approx 8$.
2. **Logarithmic utility.** Let the utility be $u(x) = \ln x$. When wealth goes from 10 to 20, the gain in utility is $\ln 20 - \ln 10 = \ln(20/10) = \ln 2 \approx 0.69$. When wealth goes from 20 to 40, the gain is $\ln 40 - \ln 20 = \ln 2 \approx 0.69$. Each doubling gives the same gain in utility; in the first, 10 lira

was added, in the second, 20 lira.

3. **Marginal utility.** $u'(x) = 1/x$: at wealth 10 the extra utility of one lira is about 0.1, at wealth 100 about 0.01. As one grows richer, the utility of an extra lira shrinks.

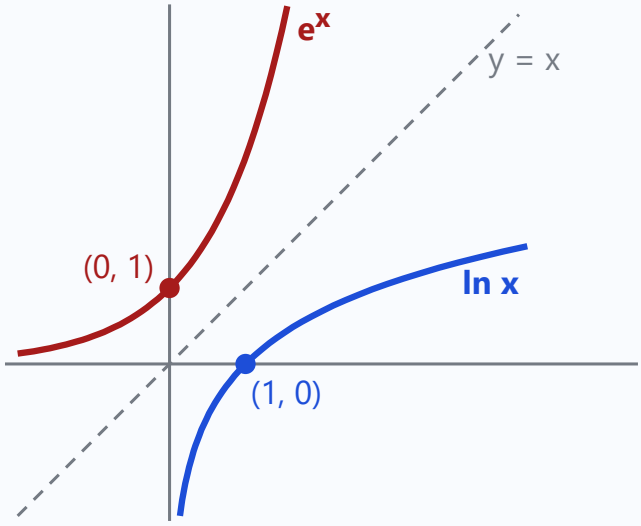


Figure 0.12: e^x (red) and $\ln x$ (blue) are mirror images of each other across the line $y = x$. e^x passes through (0, 1) and $\ln x$ through (1, 0).

Figure 0.12 shows the inverse-function relationship. A function and its inverse are symmetric about the line $y = x$: since $e^0 = 1$, the curve e^x passes through the point (0, 1), and since $\ln 1 = 0$, the curve $\ln x$ passes through (1, 0). The curve e^x is always above the x-axis; $\ln x$ is defined only for positive x.

Common mistake

Believing that $\ln(a + b) = \ln a + \ln b$. A numerical check shows the error: $\ln(2 + 6) = \ln 8 \approx 2.08$, but $\ln 2 + \ln 6 = \ln 12 \approx 2.48$. It is the logarithm of a product that turns into a sum, not that of a sum. Also, $\ln 0$ and the ln of negative numbers are undefined: e^x is never zero or negative, so the question "to what power must e be raised to give 0?" has no answer.

0.4.3 The product rule and the chain rule

The rules in 0.3.5 suffice for functions whose product we can first multiply out and write as a sum of terms. But not every function can be expanded like that. For nested functions such as $\ln(1 + 2x)$ or e^{-x}, and for long products, two more rules are needed.

Definition 0.10

- **Product rule:** $(f \cdot g)' = f' \cdot g + f \cdot g'$.
- **Chain rule:** $[f(g(x))]' = f'(g(x)) \cdot g'(x)$. First take the derivative of the outer function, leaving the inner one as it is; then multiply by the derivative of the inner one.

The product rule can be pictured as follows: $f \cdot g$ is the area of a rectangle with one side f and the other g. When x increases a little, both sides grow a little. The increase in area comes from two strips: the strip added by the growth of side f ($f' \cdot g$) and the strip added by the growth of side g ($f \cdot g'$). The tiny square in the corner is the product of the two small increments and is neglected.

The chain rule, in turn, is a transfer of rates. The inner function changes with x at rate g'; the outer function changes with its own input at rate f'. When x changes, the result changes at the product of these two rates. Figure 0.13 draws this transfer as a pipeline.

Example 0.11

1. **Product rule.** In the profit $u(q) = q \cdot (12 - q - r)$, $f = q$ and $g = 12 - q - r$. $f' = 1$, $g' = -1$ (r is constant). By the rule: $u' = 1 \cdot (12 - q - r) + q \cdot (-1) = 12 - 2q - r$. Multiplying out and then differentiating (0.3.6) gives the same result.
2. **Chain rule, logarithm.** $[\ln(1 + 2x)]'$: the outer function is $\ln$, the inner function is $g = 1 + 2x$. The outer derivative is $1/g = 1/(1 + 2x)$, the inner derivative is $g' = 2$. The result is $2/(1 + 2x)$.
3. **Chain rule, exponential.** $(e^{-x})'$: the outer derivative is e^{-x} (the exponential function is its own derivative), the inner derivative is $(-x)' = -1$. The result is $-e^{-x}$.
4. **Chain rule, square.** $[(1 - p)^2]'$: the outer function is squaring, with derivative $2(1 - p)$; the inner derivative is $(1 - p)' = -1$. The result is $2(1 - p) \cdot (-1) = -2(1 - p)$. Check: expanding $(1 - p)^2 = 1 - 2p + p^2$ gives the derivative $-2 + 2p = -2(1 - p)$ ✓.

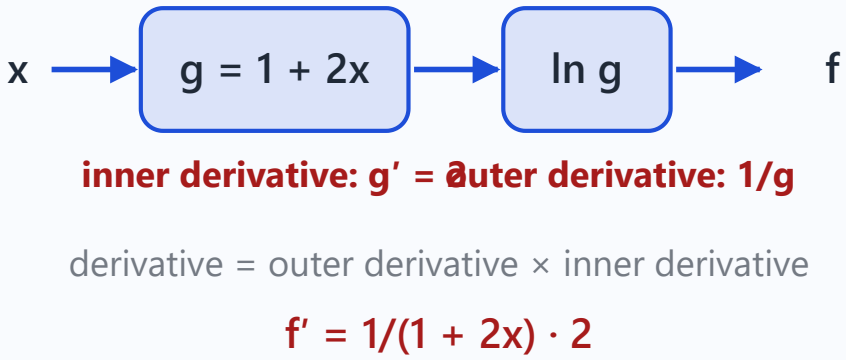


Figure 0.13: The chain rule is a pipeline: x first enters $g = 1 + 2x$ (inner derivative 2), then $\ln g$ (outer derivative $1/g$). The derivative is the product of the two rates: $f' = 1/(1 + 2x) \cdot 2$.

The chain rule is easy to check: compare the result with the one obtained by expanding the function and differentiating with known rules. This check was done in the last example. For functions that cannot be expanded (logarithm, exponential), a numerical check can be made: at $x = 0$ the derivative of $\ln(1 + 2x)$ is 2 by the formula; when x goes from 0 to 0.001 the function increases by $\ln 1.002 \approx 0.002$, so the rate of increase is about 2 ✓.

Common mistake

Forgetting the derivative of the inner function: writing $(e^{-x})' = e^{-x}$. The correct answer is $-e^{-x}$; the minus sign comes from the inner derivative. There is also a common mistake with products: writing $(f \cdot g)' = f' \cdot g'$ is wrong. For example, $(q \cdot q)' = (q^2)' = 2q$, but $f' \cdot g' = 1 \cdot 1 = 1$.

0.4.4 Partial derivatives

In a game the payoff depends not only on one player's choice but on everyone's. The firm's profit $u(q, r)$ depends both on its own quantity q and on the opponent's quantity r. For such a function of two variables the question "what is the derivative?" is ambiguous: which variable is changing? Since in a game each player can choose only her own variable, the right question is: "while the other variable stays put, how does my payoff change when my variable changes?"

Definition 0.11

The **partial derivative** $\partial u/\partial q$ of $u(q, r)$ with respect to q is the derivative taken with respect to q alone, holding r like a fixed number. $\partial u/\partial r$ is taken in the same way with q held fixed. The symbol ∂ is read "partial."

The partial derivative requires no new rule. When taking $\partial u/\partial q$, wherever r appears we act as if 4, 7, or some other constant had been written there; all the rules we know apply unchanged. In game theory each player sets the partial derivative with respect to her own variable equal to zero. This gives one equation per player, that is, a best-response formula. When these equations are solved together, we find the point where everybody is best-responding at the same time.

Example 0.12

Two firms: the first firm's profit is $u_1(q, r) = q(12 - q - r)$ and the second's is $u_2(q, r) = r(12 - q - r)$.

1. Expand the first profit: $u_1 = 12q - q^2 - qr$.
2. Hold r fixed: $\partial u_1/\partial q = 12 - 2q - r$. ($12q$ contributes 12, q^2 contributes $2q$, qr contributes r.)
3. For comparison, the other partial derivative can also be taken: with q fixed, $\partial u_1/\partial r = -q$. If the opponent produces one more unit, the price falls by one and the first firm's profit falls by q.
4. The second firm chooses its own variable r: $u_2 = 12r - qr - r^2$ and $\partial u_2/\partial r = 12 - q - 2r$.
5. Each firm sets its own partial derivative equal to zero: $12 - 2q - r = 0$ and $12 - q - 2r = 0$.
6. Subtract the second equation from the first: $(12 - 2q - r) - (12 - q - 2r) = -q + r = 0$, so $q = r$. Substitute into the first equation: $12 - 3q = 0 \Rightarrow q = 4$. The result is $q = r = 4$.
7. Check: against $r = 4$ the first firm's best quantity is $(12 - 4)/2 = 4$ (0.3.6); by symmetry the same holds for the second firm. Each firm's profit is $4 \cdot (12 - 8) = 16$.

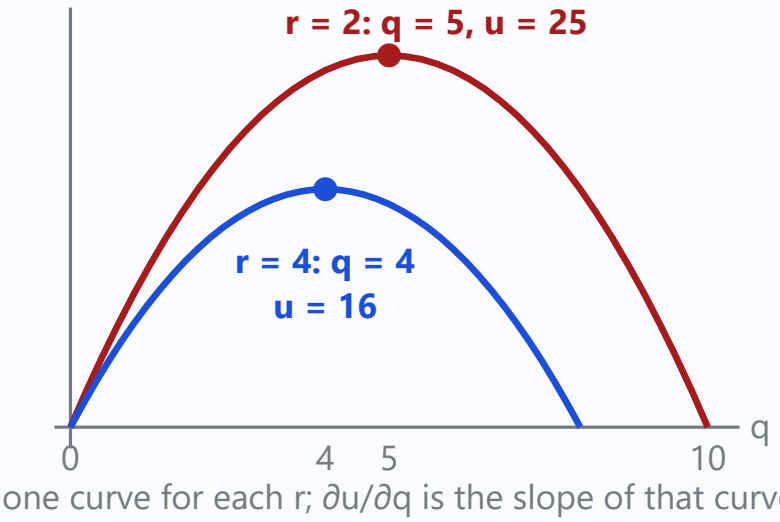


Figure 0.14: Each quantity r of the opponent gives a separate profit curve. At $r = 2$ the peak is at $q = 5, u = 25$; at $r = 4$ the peak is at $q = 4, u = 16$. $\partial u/\partial q$ is the slope of that curve.

Figure 0.14 shows the geometric meaning of the partial derivative. Each value of r gives a separate curve depending on q. $\partial u/\partial q$ is the slope of the curve selected by fixing r. The more the opponent produces, the lower the curve sits and the further left its peak moves: the firm both earns less and wants to produce less. At the point $q = r = 4$ found in the example, each firm is at the peak of its own curve, which means neither wants to change its quantity on its own. In game theory this property is called an equilibrium.

Common mistake

Setting the fixed variable to zero or differentiating it as well. $\partial(q \cdot r)/\partial q = r$; it is not 0 (r is fixed but not zero) and it is not $1 + r$ (r is not differentiated). A second mistake is to differentiate with respect to the wrong variable: the first firm cannot choose r, so its condition is not $\partial u_1/\partial r = 0$ but $\partial u_1/\partial q = 0$.

0.5 Probability and expected value

In games, uncertainty comes from two sources. The first is nature: dice, cards, the weather. The second is the opponent: what the opponent will play is not known for certain, and sometimes the opponent deliberately acts at random. In both cases the question that governs the decision is the same: "what do I win on average?" To answer it, we need to find probabilities by counting, update them when information arrives, and compute the average payoff, that is, the expected value. When the values are continuous (like every number between 0 and 1), an integral takes the place of the sum.

0.5.1 Computing probabilities by counting

If all possible outcomes of an experiment (rolling a die, drawing a card) are equally likely, the probability of an event is a ratio of counts: the number of desired outcomes divided by the number of all outcomes. A probability is always a number between 0 and 1; 0 means "never" and 1 means "certain."

Definition 0.12

- If the outcomes are equally likely: *P(event) = number of desired outcomes / number of all outcomes*.
- If two events are **independent** (neither affects the probability of the other), the probability that both occur is the product: $P(A \text{ and } B) = P(A) \cdot P(B)$.
- If two events cannot occur at the same time, the probability that one or the other occurs is the sum: $P(A \text{ or } B) = P(A) + P(B)$.
- **Complement**: the probability that an event does not occur is $P(\text{not } A) = 1 - P(A)$.

The product rule for independence is the probability version of the counting rule in 0.2.3. When two dice are rolled, each of the 6 outcomes of the first die pairs with each of the 6 outcomes of the second to form one of 36 outcomes. So the probability that both dice show 6 is $1/36 = 1/6 \cdot 1/6$. The complement rule, in turn, is very useful for "at least one" questions: instead of counting the event "at least one 6" directly, we count "no 6 at all" and subtract from 1.

Example 0.13

1. **An even number on a die.** The desired outcomes are $\{2, 4, 6\}$: three outcomes. There are six outcomes in all. $P = 3/6 = 1/2$.
2. **Two heads in two coin flips.** The flips are independent: $1/2 \cdot 1/2 = 1/4$. Check by counting: there are four equally likely outcomes (HH, HT, TH, TT) and only one is two heads.
3. **At least one 6 on two dice.** Use the complement. The probability of no 6 on one die is $5/6$. The two dice are independent, so no 6 at all has probability $(5/6) \cdot (5/6) = 25/36$. Hence $P(\text{at least one 6}) = 1 - 25/36 = 11/36$.
4. **Check by counting.** In Figure 0.15 the 36 outcomes are arranged in a table. The sixth row has 6 cells and the sixth column has 6 cells; the cell (6, 6) is in both, so it is subtracted once: $6 + 6 - 1 = 11$ orange cells.

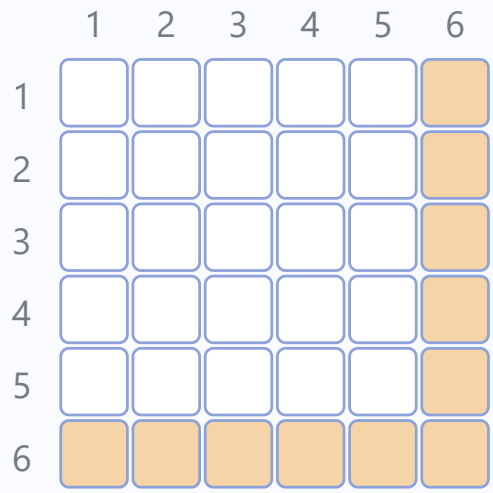


Figure 0.15: The 36 equally likely outcomes of two dice. The 11 cells containing at least one 6 are orange; the corner (6, 6) is counted once.

Common mistake

Saying $1/6 + 1/6 = 12/36$ for "at least one 6 on two dice." This sum counts the outcome (6, 6) twice; the correct answer is $11/36$. Probabilities add only if the events cannot occur together. "The first die is 6" and "the second die is 6" can both happen in the same roll. The same mistake grows with more dice: for six dice the addition method gives $6 \cdot 1/6 = 1$, that is, it claims a "certain at least one 6"; the correct answer is $1 - (5/6)^6 \approx 0.665$.

0.5.2 Probability distributions

If the outcome of an experiment is a number (the face of a die, the sum of two dice, the prize of a lottery), that number is called a **random variable** and is usually written with a capital letter: X. The table that gives a probability to each possible value of X is called a **probability distribution**.

Definition 0.13

If the values that X can take are $x_1, \ldots, x_k$, its distribution is the list of probabilities $P(X = x_j)$. Two conditions are needed for a list to be a distribution:

- every probability is between 0 and 1: $0 \le P(X = x_j) \le 1$,
- the probabilities sum to 1: $\Sigma_j P(X = x_j) = 1$.

These two conditions are used unchanged in the definition of mixed strategies (0.6.1). The probabilities a player assigns to her moves must form a distribution. Errors in computations are often caught by checking the second condition: add up the probabilities, and if the total is not 1, something is wrong.

Example 0.14

Let X be the sum of two fair dice. The values of X run from 2 to 12.

1. There are 36 equally likely outcomes (0.5.1).
2. Count the pairs that give sum 7: (1, 6), (2, 5), (3, 4), (4, 3), (5, 2), (6, 1). There are six pairs: $P(X = 7) = 6/36 = 1/6$.
3. Sum 2 occurs only with (1, 1): $P(X = 2) = 1/36$. Sum 12 also occurs only with (6, 6).
4. In general, as the sum goes from 2 to 7 the number of pairs is 1, 2, 3, 4, 5, 6; from 7 to 12 it is 5, 4, 3, 2, 1.
5. Check: $1 + 2 + 3 + 4 + 5 + 6 + 5 + 4 + 3 + 2 + 1 = 36$; the total probability is $36/36 = 1$ ✓.

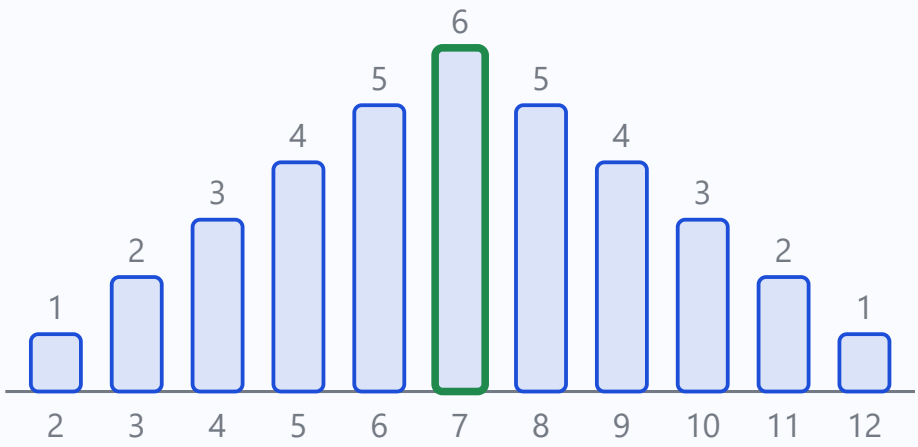


Figure 0.16: The distribution of the sum of two dice. The numbers above the bars show how many of the 36 outcomes give that sum. The most likely sum is 7 (green, 6/36).

Common mistake

Treating every sum as equally likely: "there are 11 different sums, each 1/11." What is equally likely is the *outcomes* (the pairs of dice), not the sums. Sum 7 is obtained with six different pairs; sum 2 with a single pair. The bars of different heights in Figure 0.16 are the picture of this. The counting rule for probability (0.5.1) applies only to equally likely outcomes; one must first determine what the equally likely outcomes are.

0.5.3 Conditional probability

A player gains information during the game: she sees one of the opponent's moves, receives a signal, has a card revealed. This information changes the probabilities. For example, once we learn that the card drawn from a deck is red, the probability that it is a heart rises from 1/4 to 1/2. The probability updated with the new information is called a **conditional probability**.

Definition 0.14

With $P(B) > 0$, the probability of A given that event B is known to have occurred is defined by

$$P(A \mid B) = P(A \cap B) / P(B)$$

$A \cap B$ means that both A and B occur. $P(A \mid B)$ is read "the probability of A given B."

The meaning of the formula is this: once we learn that B has occurred, the outcomes outside B are now impossible. Only the outcomes inside B remain, and in this narrowed world we look for the share of A. The denominator $P(B)$ is the size of the new world; the numerator $P(A \cap B)$ is the space that A

occupies in this world. In the card example there are 26 red cards, and 13 of them are hearts: *P(heart | red) = (13/52)/(26/52) = 1/2*.

Example 0.15

A traffic light at an intersection picks one of three situations, each with probability 1/3: (Ayşe Stop, Bora Stop), (Ayşe Stop, Bora Go), (Ayşe Go, Bora Stop). It shows each driver only her or his own signal. Ayşe sees "Stop." What is the probability that Bora has seen "Go"?

1. Let B = "Ayşe sees Stop." This event occurs in the first two situations: $P(B) = 1/3 + 1/3 = 2/3$.
2. Let A = "Bora sees Go." $A \cap B$ is only the second situation: $P(A \cap B) = 1/3$.
3. Apply the definition: $P(\text{Bora Go} \mid \text{Ayşe Stop}) = (1/3)/(2/3) = 1/2$.
4. Interpretation: when Ayşe sees "Stop," in one of the two remaining equally likely situations Bora goes and in the other he stops. For Ayşe, the probability that Bora goes has risen from 1/3 to 1/2. Figure 0.17 shows this narrowing with boxes.

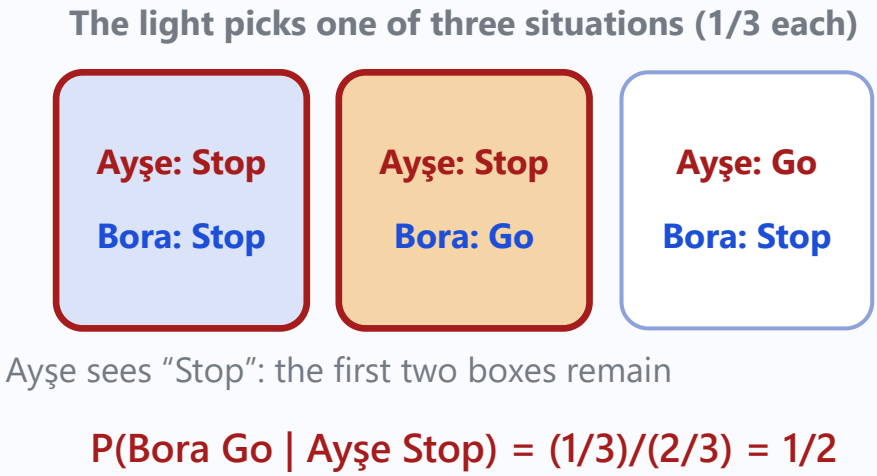


Figure 0.17: The light picks one of three situations with probability 1/3 each. When Ayşe sees "Stop," only the first two boxes remain; in one of them Bora goes: *P(Bora Go | Ayşe Stop) = (1/3)/(2/3) = 1/2*.

This example is the basis of an important idea in game theory. The light is a coordination device that prevents the drivers from colliding. Each driver decides for herself whether to follow the light's recommendation. When Ayşe sees "Stop," she knows that Bora goes half the time; with this information she can compute whether stopping is a good choice for her. Arrangements of this kind, in which the recommendations are in everyone's interest to follow, are called correlated equilibria. The whole calculation rests on conditional probability.

Common mistake

Ignoring the information ("Bora sees Go: 1/3") or mixing up the direction of the condition. $P(A \mid B)$ and $P(B \mid A)$ are generally different. In the example $P(\text{Ayşe Stop} \mid \text{Bora Go}) = 1$: if Bora goes, Ayşe certainly stops. But $P(\text{Bora Go} \mid \text{Ayşe Stop}) = 1/2$. The event that is known (right of the bar) and the event that is asked about (left of the bar) must always be written separately.

0.5.4 Bayes' rule

In games such as poker, players do not know the opponent's "type": is the opponent aggressive or calm? But they do see the opponent's moves. A large bet is a signal that the opponent is aggressive; how strong a signal it is can be found by calculation. The rule for updating a belief after an observation is **Bayes' rule**.

Theorem 0.2 (Bayes' rule)

The belief $P(A)$ before the observation is called the **prior**, and the belief $P(A \mid B)$ after observing B is called the **posterior**. If $P(B) > 0$, then

$$P(A \mid B) = P(B \mid A) \cdot P(A) / P(B)$$

The denominator is often computed from two parts: $P(B) = P(B \mid A) \cdot P(A) + P(B \mid \text{not } A) \cdot P(\text{not } A)$.

Proof. The definition of conditional probability is written in both directions. $P(A \mid B) = P(A \cap B)/P(B)$ and $P(B \mid A) = P(A \cap B)/P(A)$. Multiplying the second equality by $P(A)$ gives $P(A \cap B) = P(B \mid A) \cdot P(A)$. Putting this into the numerator of the first equality gives the rule. For the denominator the following observation suffices: B occurs either together with A or without A; these two cases cannot happen at once. So $P(B) = P(A \cap B) + P(\text{not } A \cap B)$, and each part is written with the same product rule. □

Example 0.16

Twenty percent of players are aggressive. An aggressive player makes a large bet with probability 0.8, a calm player with probability 0.2. A player has been seen making a large bet. What is the probability that this player is aggressive?

1. Prior: $P(\text{aggressive}) = 0.2$; hence $P(\text{calm}) = 0.8$.
2. Probabilities of the observation: $P(\text{big} \mid \text{aggressive}) = 0.8$, $P(\text{big} \mid \text{calm}) = 0.2$.
3. Denominator: $P(\text{big}) = 0.8 \cdot 0.2 + 0.2 \cdot 0.8 = 0.16 + 0.16 = 0.32$.
4. Numerator: $P(\text{big} \mid \text{aggressive}) \cdot P(\text{aggressive}) = 0.8 \cdot 0.2 = 0.16$.
5. Posterior: $P(\text{aggressive} \mid \text{big}) = 0.16 / 0.32 = 1/2$.

6. **Check by counting.** Imagine 100 players (Figure 0.18). 20 are aggressive, and *0.8 · 20 = 16* of them make a large bet. 80 are calm, and *0.2 · 80 = 16* of them make a large bet. Of the *16 + 16 = 32* people who make a large bet, 16 are aggressive: *16/32 = 1/2* ✓.

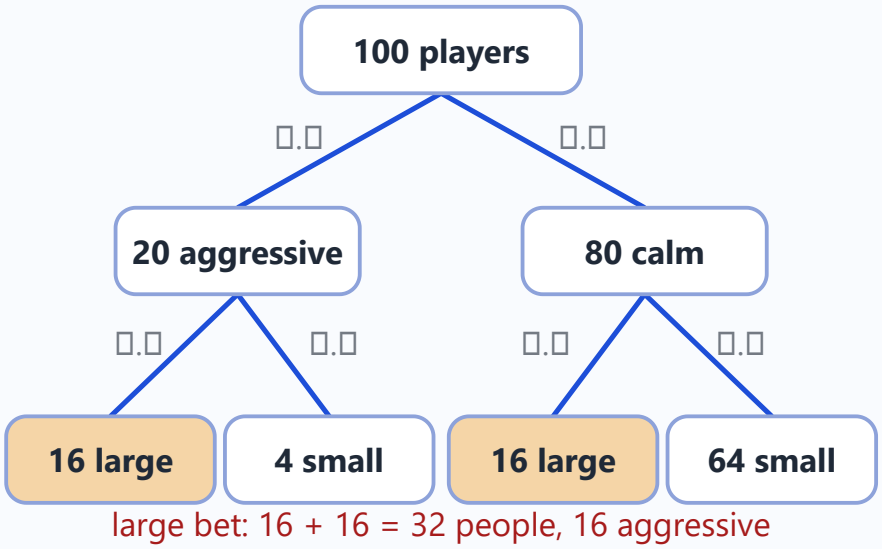


Figure 0.18: The count tree for Bayes' rule. Of 100 players, 20 are aggressive and 80 are calm. Of the 16 + 16 = 32 people who make a large bet, 16 are aggressive.

The result is surprising at first: although aggressive players make a large bet 80% of the time, the probability that someone who makes a large bet is aggressive is only 50%. The reason is the prior. Calm players are four times as numerous. Even though each of them rarely makes a large bet, there are so many of them that half of the large bets come from them. The observation has raised the belief from 0.2 to 0.5; this is a strong update, but it does not give certainty.

The count-tree method is the safest way to do Bayes computations. Instead of probabilities, imagine a group of 100 or 1000 people, find the number of people on each branch by multiplication, and at the end count only the branches consistent with the observation. Each term in the formula corresponds to a branch in the tree: the numerator is the "aggressive" branch consistent with the observation; the denominator is all the branches consistent with the observation.

Common mistake

Forgetting the prior: "Aggressive players make a large bet 80% of the time, so this player is 80% likely to be aggressive." This sentence confuses *P(big | aggressive)* with *P(aggressive | big)* (the direction error of 0.5.3). Because calm players are four times as numerous, their rare large bets also make up half of the total. If the prior is skipped, rare types always look more common than they are.

0.5.5 Expected value

The most basic tool for deciding under uncertainty is the expected value. If a game were played many times, what would be won on average per round? The answer is a weighted average in which each possible outcome is counted in proportion to its own probability.

Definition 0.15

Let X be a number whose value depends on chance. The **expected value** of X is obtained by multiplying each value x it can take by its probability and adding:

$$E[X] = \Sigma_x \, x \cdot P(X = x)$$

Why the expected value is a "long-run average" can be seen as follows. Roll a fair die 600 times. Each face comes up about 100 times. The total of all the rolls is about $1 \cdot 100 + 2 \cdot 100 + \ldots + 6 \cdot 100 = 2100$; the average per roll is $2100/600 = 3.5$. This computation is nothing other than multiplying each value by its frequency, that is, its probability, and adding.

The expected value has two useful properties. First, multiplying by a constant and adding a constant pass straight through to the expected value: $E[aX + b] = a \cdot E[X] + b$. If the prizes of a game double, the expected prize doubles; if an entry fee is deducted, the expected net gain falls by that amount. Second, the expected value of a sum is the sum of the expected values: the expected value of the sum of two dice is $3.5 + 3.5 = 7$. This second property holds even if the dice are not independent.

Example 0.17

Let X be the face shown by a fair die.

1. The values are 1, 2, 3, 4, 5, 6; each has probability 1/6 (Figure 0.19).
2. Multiply each value by its probability and add: $E[X] = 1 \cdot (1/6) + 2 \cdot (1/6) + 3 \cdot (1/6) + 4 \cdot (1/6) + 5 \cdot (1/6) + 6 \cdot (1/6)$.
3. Take the common factor 1/6 outside: $E[X] = (1 + 2 + 3 + 4 + 5 + 6)/6 = 21/6 = 3.5$.



Figure 0.19: The distribution of a fair die: each face has probability 1/6. The expected value *E[X] = 21/6 = 3.5* is not under any bar.

In game theory the expected value is everywhere. A player computes the expected payoff of each of her moves against the opponent's probabilistic behavior and chooses the largest (0.6.2). A mechanism designer computes the participants' expected payments. All of these computations apply the same formula: value times probability, then add.

Common mistake

Taking the expected value to be "the most likely outcome" or "the outcome that will certainly occur." A die never shows 3.5; 3.5 is only the average of many rolls. A lottery whose payoff is 0 or 100 can have expected value 30 (if it pays 100 with probability 0.3), but the lottery never pays 30. The expected value describes not the result of a single play but the average of repetition.

0.5.6 Code trace: the expected value of a lottery

A lottery ticket wins 100 TL with probability 0.05, 20 TL with probability 0.25, and nothing with probability 0.70. The ticket costs 12 TL. The code below is the same as the code in the slides and applies the expected-value formula with a loop.

```
prize = [100, 20, 0]          # TL
prob = [0.05, 0.25, 0.70]
E = 0
for i in range(3):
    E = E + prize[i] * prob[i]
print("E =", E)
print("net =", E - 12)  # ticket 12 TL
```

The structure of the code is a one-to-one counterpart of the Σ sum:

- The lists `prize` and `prob` are matched by index: index 0 is the largest prize (100 TL, 0.05).
- `E = 0`: the accumulator starts at zero. It is the "nothing added yet" state of Σ.
- At each step of the loop one term, `prize[i] * prob[i]`, is added to the accumulator.
- The last two lines print the expected prize and the net expected gain after the ticket price is deducted.

i	prize[i]	prob[i]	Term	E (after the step)
–	–	–	–	0
0	100	0.05	5	5
1	20	0.25	5	10
2	0	0.70	0	10

Table 0.2: Trace of the lottery code.

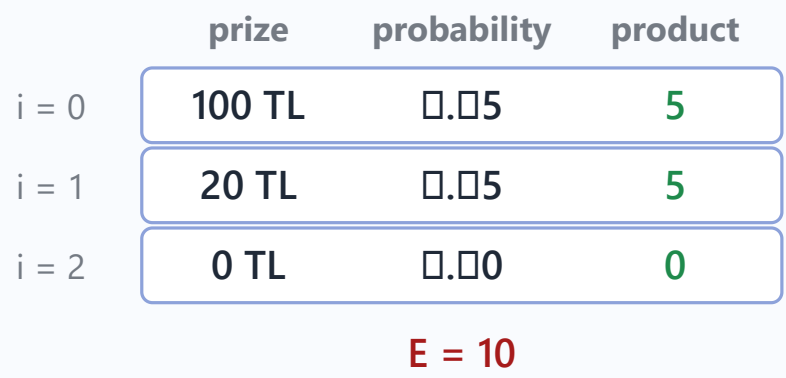


Figure 0.20: The final state of the trace: the three terms are 5, 5, and 0; the accumulator is $E = 10$. The probabilities sum to $0.05 + 0.25 + 0.70 = 1$.

Table 0.2 shows the term added at each step and the new value of the accumulator, and Figure 0.20 shows the final state. The notable step of the trace is the third term. The prize of a blank ticket is 0, and $0 \cdot 0.70 = 0$ does not change the accumulator. But this row is not superfluous: its probability is 0.70, and it must be counted for the distribution to sum to 1. Checking the sum of the probabilities shows that no outcome has been forgotten.

The expected prize of one ticket is 10 TL. Since the ticket costs 12 TL, the net expected gain is $10 - 12 = -2$ TL. Someone who plays many times loses on average 2 TL per ticket. The same result also follows from the linearity of the expected value: the net gain is $X - 12$ and $E[X - 12] = E[X] - 12 = -2$. When the code is run it prints two lines: `E = 10.0` and `net = -2.0`. Python writes decimal numbers with a point (`10.0`). In this example the result comes out exact; with other probabilities a number very close to it, such as `9.999999999999998`, may appear because of floating-point rounding. This does not mean the computation is wrong; it shows that some decimal numbers cannot be represented exactly in binary.

0.5.7 Is this dice game worth playing?

The most direct use of expected value is to decide whether a game is worth entering. The computation can be organized neatly in a table; the question of a "fair price" is answered with an equation in one unknown.

Example 0.18

A game at a fair: you pay 10 TL and a die is rolled. If it shows 6 you receive 40 TL, if it shows 5 you receive 10 TL; on any other number you receive nothing.

1. The possible prizes and their probabilities are written in a table (Table 0.3). Since 1, 2, 3, and 4 give the same prize (0), they are merged into one row; their probabilities add: $4/6$.
2. For each row compute $x \cdot P(x)$: $40/6$, $10/6$, and 0.
3. The expected prize is $50/6 \approx 8.33$ TL.
4. Since the entry fee is 10 TL, the net expected gain is $50/6 - 10 = -10/6 \approx -1.67$ TL. If risk is ignored, playing is not profitable.

Die	Prize x	P(x)	x · P(x)
6	40	1/6	40/6
5	10	1/6	10/6
1, 2, 3, 4	0	4/6	0
Total		6/6 = 1	50/6 ≈ 8.33

Table 0.3: The expected-value table of the dice game.

What would make the game fair? A game is called *fair* if its net expected gain is zero. Let the prize for 6 be an unknown y, with the other prizes unchanged. The expected prize is $y/6 + 10/6 = (y + 10)/6$. For fairness this must equal the entry fee:

$$(y + 10)/6 = 10 \Rightarrow y + 10 = 60 \Rightarrow y = 50$$

If the prize for 6 were 50 TL, the game would be fair. The same question can be asked another way: if the prizes stay the same, what should the entry fee be? The answer is the expected prize itself, about 8.33 TL. This is an equation in one unknown; equations in two unknowns are treated in 0.9.2.

Common mistake

Deciding with "my chance of winning is 2/6, not bad." The probability of winning alone is not enough; the size of the prizes and the entry fee count as well. If the prize for 6 were 1000 TL, the game would be very profitable with the same 2/6 chance; if it were 11 TL, it would be very

costly. The decision is made by the net expected value. (A risk-averse person may reject even a game with positive net expected value; this preference is modeled with utility functions.)

0.5.8 The continuous uniform distribution U[0, 1]

So far the random variables have taken one of a few values. Some quantities, however, can be any number in an interval: the value a buyer places on a good, a bid, a waiting time. The most common assumption in auction models is that the opponents' values are spread "equally likely" between 0 and 1. This distribution is called the **uniform distribution**.

Definition 0.16

The notation $X \sim U[0, 1]$ is read "X is uniformly distributed between 0 and 1": every value in this interval is equally likely. Its graph is a rectangle of height 1; this height is called the **probability density**: $f(x) = 1$ $(0 \le x \le 1)$. Probability is the *area* under the density:

$$P(a \le X \le b) = b - a \text{ and in particular } P(X \le x) = x.$$

In a continuous distribution the probability of a single point is zero. This is odd at first: the value 0.5 may well occur, so how can its probability be zero? The explanation is that the interval contains infinitely many values. If each were given a positive probability, the total would be infinite. So probability is given not to points but to intervals; the probability of an interval is its length. A point, whose length is zero, has probability zero. One consequence is that $P(X \le 0.6)$ and $P(X < 0.6)$ are the same.

Example 0.19

In an auction the opponent's value is $X \sim U[0, 1]$. The opponent bids her value.

1. **One opponent.** A bid of 0.6 wins if the opponent's value is below 0.6: $P(X \le 0.6) = 0.6$ (Figure 0.21). Area: base 0.6 × height 1.
2. **Two independent opponents.** To beat both, two events must occur together. By independence, $0.6 \cdot 0.6 = 0.36$.
3. **An interval.** $P(0.2 \le X \le 0.5) = 0.5 - 0.2 = 0.3$.
4. **The mean.** By symmetry the mean is the middle of the interval: $E[X] = 1/2$. This result is also found with an integral in 0.5.9.

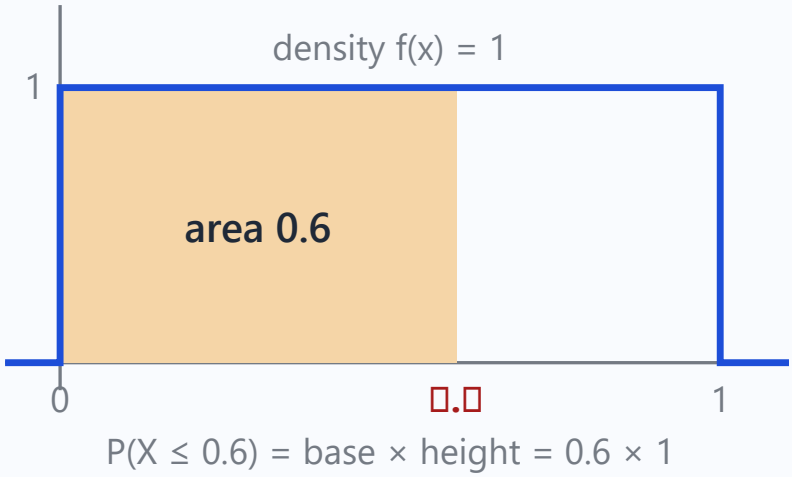


Figure 0.21: The density of *U[0, 1]*: a rectangle of height 1. *P(X ≤ 0.6)* is the orange area: base × height = *0.6 × 1 = 0.6*.

The effect of the product rule in the second step grows quickly with the number of opponents. The probability of beating all n independent opponents with a bid of 0.6 is 0.6^n: 0.6 with 1 opponent; 0.36 with 2; about 0.078 with 5. As the number of opponents grows, a higher bid is needed to win. Many computations in auction theory rest on this simple observation.

Common mistake

Mistaking the height of the density (1) for a probability, or giving *P(X = 0.5)* a positive value. In a continuous distribution a probability is an area: base (interval) × height. The density can even exceed 1: the density of *U[0, 0.5]* is 2, because the area above a base of length 0.5 must be 1. A density is not a probability but "probability per unit length."

0.5.9 Integrals: the area under a curve

In the uniform distribution the areas are rectangles and so are easy to compute. But in computations such as expected values we need areas under sloped or curved lines. Finding the area under a curve is called **integration**. The integral is the continuous version of the Σ sum.

Definition 0.17

$\int_a^b f(x)\,dx$ is the area under the graph of f between $x = a$ and $x = b$. The area is divided into thin rectangles and added; as the rectangles get thinner, the total approaches the integral. dx stands for the very small width of the rectangles.

The idea in the definition is directly connected to Σ. If the interval is divided into n equal pieces, each piece has width $(b - a)/n$. On each piece a rectangle is placed whose height is the value of f on that piece. The sum of the areas of the rectangles is a Σ. As n grows, the rectangles fit the curve better and the total approaches the true area. The summation sign of Σ has turned into an elongated S (∫) in the integral; this S is the initial of the Latin word "summa" (sum).

Example 0.20

1. **With geometry.** $\int_0^1 x\,dx$ is the area under the line $y = x$ between 0 and 1. This area is a triangle with base 1 and height 1: $1 \cdot 1 / 2 = 1/2$.
2. **With rectangles.** Divide the interval into four pieces, each of width 1/4. Take the height at the middle of each piece: 1/8, 3/8, 5/8, 7/8 (Figure 0.22). The total is $(1/8 + 3/8 + 5/8 + 7/8) \cdot 1/4 = (16/8) \cdot 1/4 = 1/2$. Because the graph is a straight line, the midpoint method gives the exact result here.
3. **General rule.** By the same triangle argument, $\int_0^b x\,dx = b \cdot b / 2 = b^2/2$.
4. **Probability is an area.** For $U[0, 1]$, $P(X \le b) = \int_0^b 1\,dx = b$: a rectangle of height 1 and base b.
5. **Expected value is an area.** $E[X] = \int_0^1 x \cdot 1\,dx = 1/2$. This is the continuous counterpart of the formula $E[X] = \Sigma\, x \cdot P(x)$: an integral in place of the sum, and density times a small width in place of probability.

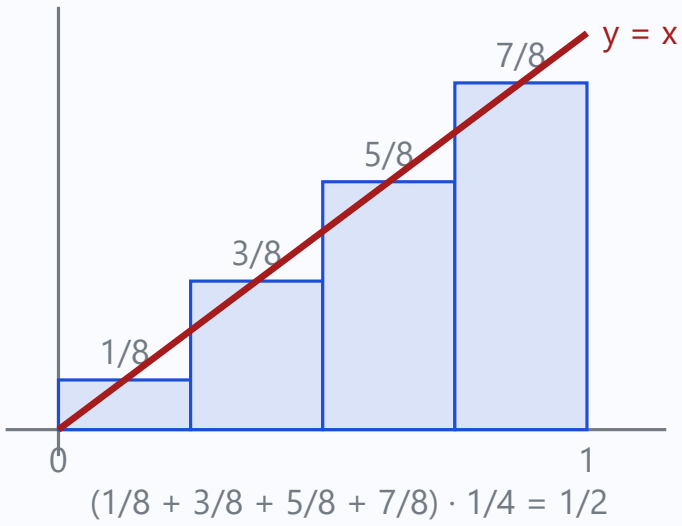


Figure 0.22: The area under $y = x$ using four rectangles of width 1/4: $(1/8 + 3/8 + 5/8 + 7/8) \cdot 1/4 = 1/2$. This is the value of $\int_0^1 x\,dx$.

In these notes the integral is needed only as the idea of area; techniques of integration are not required. The areas we meet are rectangles and triangles. What matters is to see that probability and expected value are areas in the continuous case.

Common mistake

Mistaking the integral for the height of the curve at a point. The result of $\int_0^1 x\,dx$ is not the height at $x = 1$ (which is 1) but the accumulated area, 1/2. A similar mistake is forgetting to divide by 2 in the area of a triangle: for $\int_0^2 x\,dx$ the right answer is $2 \cdot 2 / 2 = 2$, not $2 \cdot 2 = 4$. $E[X] = \int x \cdot f(x)\,dx$ is the continuous counterpart of $E[X] = \Sigma\, x \cdot P(x)$; the density factor $f(x)$ must not be dropped.

0.5.10 Expected value and tail probability

There is a second way to compute an expected value. This way is often used in revenue and payment computations in auction theory. The idea is this: instead of looking at the values of X directly, look at the probability that X exceeds each threshold.

Definition 0.18

$P(X > t)$ is the probability that X exceeds the threshold t. It is called the **tail probability**. If $X \geq 0$, the expected value is the area under the tail-probability curve:

$$E[X] = \int_0^\infty P(X > t)\,dt$$

The sum form, for X with values 0, 1, 2, ...: $E[X] = P(X \geq 1) + P(X \geq 2) + P(X \geq 3) + \ldots$

Why the sum form is correct can be seen by counting. Consider an outcome with $X = 3$. This outcome belongs to all three of the events "$X \geq 1$," "$X \geq 2$," and "$X \geq 3$," but not to "$X \geq 4$." In the sum of the tails this outcome is counted exactly three times, that is, as many times as its own value. In general, every outcome with value x is counted x times in the tail sum. So the sum of the tails equals $\Sigma\, x \cdot P(X = x)$, the expected value. The integral form is the continuous version of the same idea.

Example 0.21

1. **Uniform distribution.** For $X \sim U[0, 1]$ and $0 \leq t \leq 1$, $P(X > t) = 1 - t$; for $t > 1$ it is 0.
2. The tail curve is a line descending from (0, 1) to (1, 0) (Figure 0.23). The area under it is a triangle with base 1 and height 1: $E[X] = \int_0^1 (1 - t)\,dt = 1/2$ ✓. This agrees with the result in 0.5.9.
3. **Die.** $P(X \geq 1) = 6/6$, $P(X \geq 2) = 5/6$, ..., $P(X \geq 6) = 1/6$.
4. Sum: $(6 + 5 + 4 + 3 + 2 + 1)/6 = 21/6 = 3.5$ ✓. This also agrees with the result in 0.5.5.

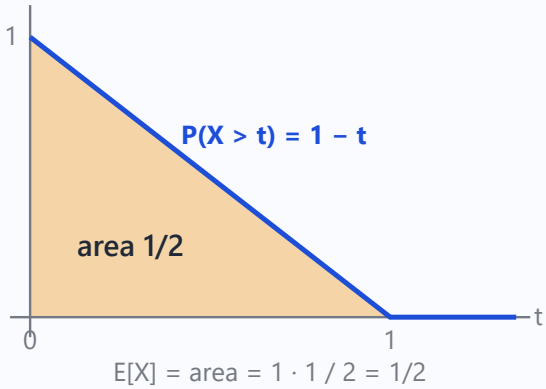


Figure 0.23: The tail probability $P(X > t) = 1 - t$ for $U[0, 1]$. The area of the orange triangle under it is $1 \cdot 1 / 2 = 1/2 = E[X]$.

Why is the tail formula needed? In some problems the distribution of X is known not directly but in the form of threshold probabilities. For example, the probability of exceeding the highest value among two opponents is easily found with the product rule (0.5.8); the shortest way from this probability to the expected value is the tail formula.

Common mistake

Mistaking the area under the density for the expected value. For $U[0, 1]$ the area under the density is 1 (total probability); $E[X] = 1/2$ is the area under the curve $P(X > t)$. The two curves are different. Also, the formula is valid only for $X \geq 0$; for a variable that can take negative values, an additional term is needed.

0.6 Mixed strategies

In Rock-Paper-Scissors, being predictable loses. Someone who always plays Rock loses every round once the opponent notices. The remedy is to leave the choice of move to chance. This idea combines probability with game theory: the player now chooses not a move but a probability distribution over moves. When the opponent plays this way, each move is evaluated by its expected payoff. A soft rule that turns values into probabilities (softmax) can then take the place of the sharp argmax choice.

0.6.1 The idea of a mixed strategy

A player makes a single move each round. But before choosing it she can roll a die or flip a coin and let the result decide her move. The opponent then cannot know in advance which move the player will make; at most he can learn with what probabilities she plays.

Definition 0.19

A **mixed strategy** is a probability distribution that a player places over her own actions. In Rock-Paper-Scissors a mixed strategy is written with three numbers: (p_R, p_P, p_S); each number is between 0 and 1 and they sum to 1.

Playing a single action with certainty (a distribution with one probability equal to 1 and the others 0) is called a **pure strategy**. The set of actions given positive probability is called the **support** of the strategy.

A mixed strategy adds no new move; it combines the existing moves with a randomizing device. Every pure strategy is a special case of a mixed strategy: *(1, 0, 0)* means "always Rock." Mixed strategies thus form a larger set that includes the pure strategies as well.

Example 0.22

1. **An even mixture with a die.** Roll a die: 1–2 gives Rock, 3–4 gives Paper, 5–6 gives Scissors. Each move corresponds to two of the six outcomes: *(2/6, 2/6, 2/6) = (1/3, 1/3, 1/3)*.
2. **A biased mixture with two coins.** Flip a coin; if it shows heads, play Rock. If it shows tails, flip a second coin; heads gives Paper, tails gives Scissors. The probability of Rock is 1/2. Paper requires two independent events (tails, then heads): *1/2 · 1/2 = 1/4*. Scissors is likewise 1/4. The result is *(1/2, 1/4, 1/4)*; the total is *1/2 + 1/4 + 1/4 = 1* ✓.
3. **A pure strategy.** "Always Rock" is the mixed strategy *(1, 0, 0)*. Its support is only {Rock}.

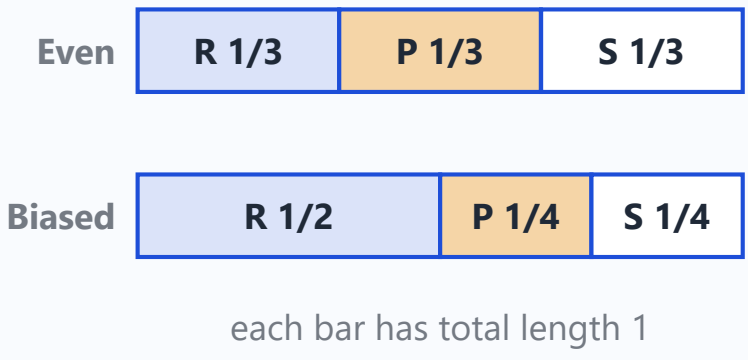


Figure 0.24: Two mixed strategies as strips: the even mixture *(1/3, 1/3, 1/3)* and the biased mixture *(1/2, 1/4, 1/4)*. Each strip has total length 1; the lengths of the pieces are the probabilities.

The strip picture in Figure 0.24 also gives a way to carry out a mixed strategy. Draw a uniform random number between 0 and 1 (0.5.8) and see which piece of the strip it falls in. In the biased mixture, play Rock if the number is below 0.5, Paper if it is between 0.5 and 0.75, and Scissors if it is above 0.75. Since the lengths of the pieces equal the probabilities, each move is chosen with the right probability. Computer programs implement mixed strategies in exactly this way.

Common mistake

Thinking a mixed strategy means "taking turns" (Rock, Paper, Scissors, Rock, ...). In this pattern each move appears with frequency 1/3 in the long run, but the pattern is predictable: an opponent who knows that Paper follows Rock plays Scissors and wins every round. In a mixed strategy a single pure move is played each round; which one is chosen is drawn anew in every round, independently of the earlier ones.

0.6.2 Expected payoff against a mixed strategy

If the opponent plays a mixed strategy, the payoff of a move is not certain; it depends on chance. So each move is evaluated by its expected payoff under the opponent's probabilities. The computation is a direct application of the expected-value formula in 0.5.5: each number in the player's row of the payoff matrix is multiplied by the probability that the opponent plays the corresponding column, and the products are added.

Example 0.23

Bora plays *(1/2 Rock, 1/4 Paper, 1/4 Scissors)*. We look for the expected payoff of each of Ayşe's pure moves.

1. **Ayşe plays Rock.** Ayşe gets 0 if Bora plays Rock, −1 if Paper, +1 if Scissors. Expected: $0 \cdot 1/2 + (-1) \cdot 1/4 + 1 \cdot 1/4 = 0$.
2. **Ayşe plays Paper.** +1 against Rock, 0 against Paper, −1 against Scissors. Expected: $1/2 + 0 - 1/4 = 1/4$.
3. **Ayşe plays Scissors.** −1 against Rock, +1 against Paper, 0 against Scissors. Expected: $-1/2 + 1/4 + 0 = -1/4$.
4. **Decision.** $\text{argmax}\{0, 1/4, -1/4\} = \text{Paper}$. Against an opponent who favors Rock, Paper earns on average 1/4 point per round.

Ayşe ↓ / Bora →	Rock (1/2)	Paper (1/4)	Scissors (1/4)	Expected payoff
Rock	0	−1	+1	0 − 1/4 + 1/4 = 0
Paper	+1	0	−1	**1/2 + 0 − 1/4 = 1/4**

Ayşe ↓ / Bora →	Rock (1/2)	Paper (1/4)	Scissors (1/4)	Expected payoff
Scissors	−1	+1	0	−1/2 + 1/4 + 0 = −1/4

Table 0.4: Ayşe's payoffs and the expected value of each row when Bora plays (1/2, 1/4, 1/4).

This two-step computation, first the expected value of each row and then the argmax of these, is the most basic computation in game theory. The result is called the **best response** (0.8). Table 0.4 is a tidy way of writing this computation: the last column shows the expected value of each row, and the green cell shows the best response.

Ayşe can also mix. If Ayşe plays (x_R, x_P, x_S), her expected payoff is the average of the row expectations weighted by these probabilities. For example, if Ayşe plays $(1/3, 1/3, 1/3)$, her payoff is $1/3 \cdot 0 + 1/3 \cdot 1/4 + 1/3 \cdot (-1/4) = 0$; this is worse than pure Paper. The expected payoff of a mixture is the average of the expected payoffs of the mixed moves, and therefore cannot exceed the payoff of the best pure move. This observation is taken up again in matrix language in 0.7.3.

Common mistake

Looking only at the opponent's most frequent move ("Bora plays Rock most often, so Paper"). If Bora played $(0.4 \text{ Rock}, 0.25 \text{ Paper}, 0.35 \text{ Scissors})$, Paper, which beats Rock, would bring $0.4 - 0.35 = 0.05$. Rock brings $-0.25 + 0.35 = 0.10$; Scissors is $-0.4 + 0.25 = -0.15$. The best move is Rock, because Scissors is also played with high probability and Rock beats Scissors. No decision should be made without computing the expected value of every row.

0.6.3 Exponential weights: logit and softmax

Choosing the best (argmax) once the expected payoffs have been computed is a sharp rule: even a small difference changes the choice completely. Real people and learning algorithms often behave more softly: they choose the good option more often and the bad option more rarely, but never rule the bad one out entirely. The rule that models this behavior is called **softmax**.

Definition 0.20

For options with values $v_1, \ldots, v_k$, the softmax rule gives the probabilities

$$P(i) = e^{\lambda v_i} / \Sigma_j e^{\lambda v_j}$$

$\lambda \geq 0$ is the **sensitivity**. With two options the rule is called the **logit** and is written as

$$P(1) = 1 / (1 + e^{-\lambda(v_1 - v_2)})$$

The structure of the rule is simple. The value of each option is first multiplied by λ and then used as the exponent of e. These exponentials are always positive. Finally each is divided by the total, so the results lie between 0 and 1 and sum to 1. If $\lambda > 0$, the option with the larger value gets a larger exponential and hence a larger probability.

How the two-option case comes out of softmax can be seen as follows: $P(1) = e^{\lambda v_1}/(e^{\lambda v_1} + e^{\lambda v_2})$. Divide the numerator and the denominator by $e^{\lambda v_1}$. The numerator becomes 1; in the denominator what remains is $1 + e^{\lambda v_2}/e^{\lambda v_1} = 1 + e^{-\lambda(v_1 - v_2)}$ (exponent rule: $e^a/e^b = e^{a-b}$). This formula shows plainly that the probability depends only on the *difference of the values*.

The two extreme cases of λ give the meaning of the rule. When $\lambda = 0$, all the exponentials are $e^0 = 1$ and every option gets equal probability: the values are ignored completely. As λ grows, the exponential of the largest value becomes much larger than the others and the probability concentrates on the argmax (if several options share the largest value, the probability is split equally among them). Softmax is an adjustable bridge between random choice and argmax.

Example 0.24

The expected payoffs are as in 0.6.2: Rock 0, Paper 1/4, Scissors −1/4, and $\lambda = 4$.

1. The exponents $\lambda \cdot v$: $4 \cdot 0 = 0$, $4 \cdot 1/4 = 1$, $4 \cdot (-1/4) = -1$.
2. The exponentials: $e^0 = 1$, $e^1 \approx 2.718$, $e^{-1} \approx 0.368$.
3. The total: $1 + 2.718 + 0.368 \approx 4.086$.
4. The probabilities: $1/4.086 \approx 0.24$, $2.718/4.086 \approx 0.67$, $0.368/4.086 \approx 0.09$. The result is $P \approx (0.24, 0.67, 0.09)$.
5. Check: $0.24 + 0.67 + 0.09 = 1$ ✓. The best move, Paper, is chosen most often and the worst move, Scissors, least often; none is zero.

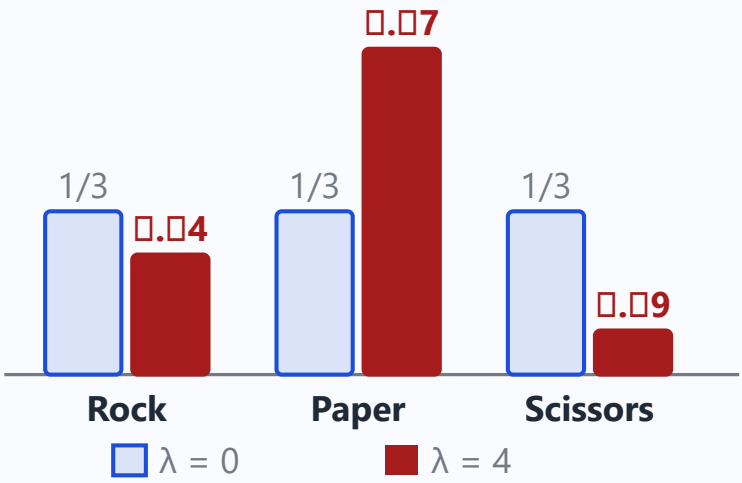


Figure 0.25: Softmax probabilities. At $\lambda = 0$ (light bars) each move has 1/3; at $\lambda = 4$ (dark bars) Rock has 0.24, Paper 0.67, Scissors 0.09.

Figure 0.25 shows the effect of the sensitivity. If λ were raised from 4 to 40, the exponents would be 0, 10, and −10, and the probability of Paper would rise to about 0.99995: the rule would become almost argmax. Softmax is used in behavioral game theory to model "players who make small mistakes" and in machine learning to model the balance between exploration and exploitation. The steady states in which players respond to one another with softmax are called quantal response equilibria (McKelvey and Palfrey, 1995).

Common mistake

Dividing the values directly by their total: $v_i \,/\, \Sigma\, v_j$. Values can be zero or negative. In the example $0 + 1/4 - 1/4 = 0$, which cannot be divided by, and the probability of Scissors would also come out negative. e^x is used because it is always positive. Also, since $e^a/e^b = e^{a-b}$, adding the same number to all the values does not change the probabilities: if the values were 10, 10.25, 9.75 instead of 0, 1/4, −1/4, softmax would give the same result.

0.7 The payoff matrix

All the information of a two-player game fits in a single table: the rows are one player's options, the columns the other's; each cell holds the two players' payoffs in that profile. This table is called a **payoff matrix**. Reading a matrix correctly is the most frequently used skill in game theory. For mixed strategies the expected payoff is also written with a matrix product.

0.7.1 Reading a payoff matrix

Definition 0.21

In the payoff matrix of a two-player game, the rows are chosen by the **row player** and the columns by the **column player**. The cell where the two choices meet is the outcome of the game. In the pair *(a, b)* in a cell, *a* is the payoff of the row player and *b* is the payoff of the column player. If *a + b = 0* in every cell, the game is zero-sum.

The definition contains two separate pieces of information. The first is *who chooses what*: the row player chooses only the row, and the column player only the column. Neither player can choose the cell alone. The second is *whom the numbers in the cell belong to*: the first number to the row player, the second to the column player. This order is the same as the order in the Cartesian product (0.2.1): profile (row choice, column choice), payoff (row payoff, column payoff).

Example 0.25

Guess the Card. Ayşe holds a red or a black card; Bora guesses the color. If Bora guesses red correctly he takes 3 chips from Ayşe, if he guesses black correctly he takes 1 chip; if he is wrong he gives Ayşe 1 chip.

1. Ayşe is the row player (Red, Black); Bora is the column player (a guess of "Red" or "Black").
2. (Red, Red): Ayşe held red and Bora guessed right. Ayşe gives 3, Bora gets 3: *(−3, 3)*.
3. (Black, Black): Bora guessed black correctly; Ayşe gives 1: *(−1, 1)*.
4. (Red, Black) and (Black, Red): Bora was wrong; Ayşe gets 1: *(1, −1)*.
5. In every cell *a + b = 0*: the game is zero-sum.

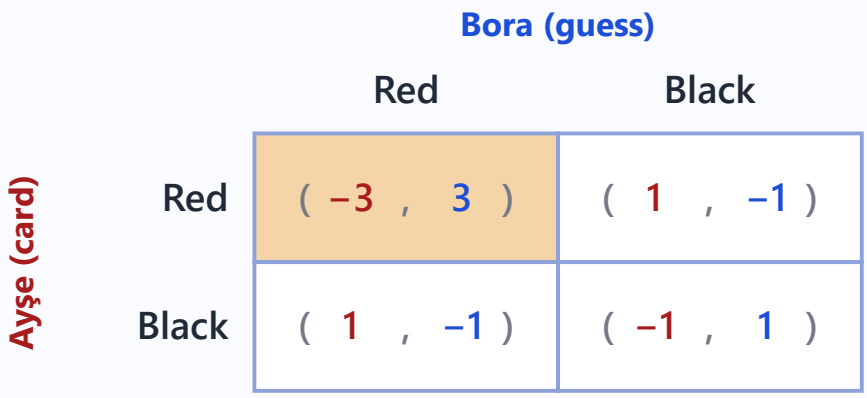


Figure 0.26: The payoff matrix of Guess the Card. Rows are Ayşe's card, columns are Bora's guess. The orange cell (Red, Red) = (−3, 3): Ayşe gives 3, Bora gets 3.

In zero-sum games the second number is always the negative of the first, so often only the row player's payoffs are written. For Guess the Card this is the matrix *A = [[−3, 1], [1, −1]]*: the first row is Red, the second row is Black; the first column is the guess "Red," the second column the guess "Black." This short form is used in 0.7.3.

Common mistake

Taking the number *b* to be the row player's payoff, or thinking that Ayşe alone chooses the outcome. Ayşe chooses only the row; Bora's column determines which cell is reached. The sentence "Ayşe can get at most 1, so she picks 1" is wrong: to get 1, Bora must guess wrong, and that is up to Bora.

0.7.2 Practice in reading a matrix

The reading rules are reinforced with a few questions. The questions concern the Guess the Card matrix (Figure 0.26). The answer to each question also states which row or column and which number are being looked at.

Example 0.26

1. **If Ayşe holds Black and Bora says "Red," who gets how many chips?** Look at the cell where the Black row meets the "Red" column: *(1, −1)*. Bora is wrong; Ayşe gets 1 chip and Bora gives 1.
2. **What payoffs can Bora get when Ayşe holds Black?** Look at the second numbers in the Black row: −1 (if he says "Red") and 1 (if he says "Black").
3. **If Bora says "Red" every round, which card should Ayşe hold?** Compare the first numbers in the "Red" column: Red −3, Black 1. Since *1 > −3*, Black.
4. **What are the largest and smallest payoffs Ayşe can get in the game?** All the first numbers: −3, 1, 1, −1. The largest is 1, the smallest −3.

The method of the third question is especially important: the opponent's choice is fixed (one column is selected), the player compares her own numbers (the first numbers) and picks the largest. This is the definition of the best response (0.8.1). The fourth question gives the limits of what Ayşe can experience in the game; which outcome within these limits she gets depends on Bora's choice.

0.7.3 Expected payoff is a matrix–vector product

When both players play mixed strategies, to compute the expected payoff the payoff of each cell is multiplied by the probability of landing in that cell and the products are added. As games grow, this computation becomes crowded. Matrix notation splits the computation into two orderly steps and allows it to be done on a computer in a single operation.

Definition 0.22

A mixed strategy is a **probability vector**: $x = (x_1, \ldots, x_m)$, with every $x_i \geq 0$ and $\Sigma\, x_i = 1$. Let A be the row player's $m \times n$ payoff matrix and y the column player's probability vector.

- Ay: each row of A is multiplied by y and added; each of the resulting m numbers is that row's expected payoff against y: $(Ay)_i = \Sigma_j A_{ij} y_j$.
- $x^T Ay$: these row expectations are weighted by the probabilities of x and added: $x^T Ay = \Sigma_i x_i (Ay)_i$. This is the row player's expected payoff.

The symbol ^{T} indicates that x is written as a row.

These two steps are the same computation as in 0.6.2. There, the expected value of each row was found first (the last column of Table 0.4); this is Ay. Then Ayşe's mixture weighted those numbers; this is $x^T Ay$. The matrix notation teaches nothing new; it merely expresses the same computation, for games of any size, in the same short form.

Example 0.27

In Guess the Card, Ayşe's payoff matrix is $A = [[-3, 1], [1, -1]]$. Ayşe plays $x = (1/2, 1/2)$ and Bora plays $y = (1/4, 3/4)$.

1. **First row (Red).** $-3 \cdot 1/4 + 1 \cdot 3/4 = -3/4 + 3/4 = 0$.
2. **Second row (Black).** $1 \cdot 1/4 + (-1) \cdot 3/4 = 1/4 - 3/4 = -1/2$.
3. So $Ay = (0, -1/2)$.
4. **Weighting.** $x^T Ay = 1/2 \cdot 0 + 1/2 \cdot (-1/2) = -1/4$. The layout of the computation is drawn in Figure 0.27.
5. **Interpretation.** Ayşe loses on average 1/4 chip per round; Bora (zero-sum game) gains 1/4. The largest element of Ay is 0 and belongs to the Red row: if Ayşe always held Red against this y, her loss would fall to 0.

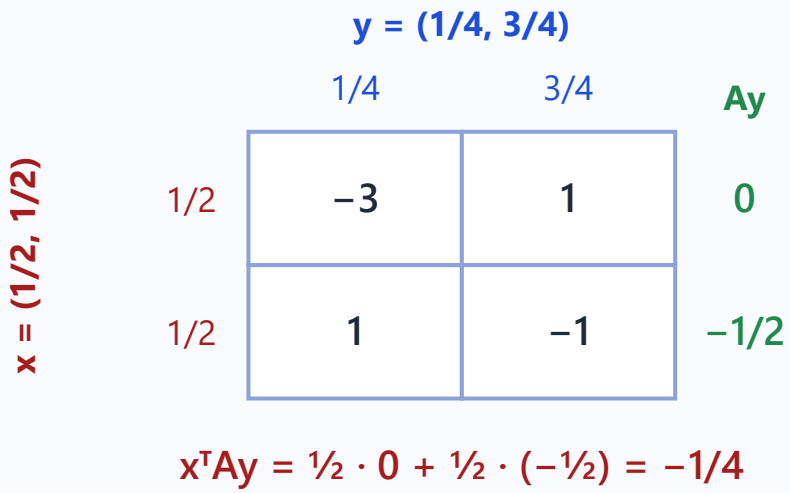


Figure 0.27: The computation of x^TAy. The column probabilities $y = (1/4, 3/4)$ are at the top and the row probabilities $x = (1/2, 1/2)$ on the left. The column on the right is $Ay = (0, -1/2)$; the result is $\frac{1}{2} \cdot 0 + \frac{1}{2} \cdot (-\frac{1}{2}) = -1/4$.

The last step of the example shows a general rule: x^TAy is a weighted average of the elements of Ay, and a weighted average cannot exceed the largest element. So if the opponent's strategy is known, a player can do as well with a pure best response as with any mixture. The purpose of mixing is not to win more against the opponent but to prevent the opponent from exploiting us. This difference is the essence of the idea of indifference (0.9.3).

Common mistake

Finding Ay and stopping there, or using x and y the wrong way round. y weights the columns (Ay: the expectation of each row); x weights these row expectations. If done the other way (the rows weighted with y), the dimensions do not match or wrong numbers come out. The largest element of Ay gives the row player's best pure response to y.

0.8 Best response

All the tools of the previous sections come together in a single concept: the best response. The payoff function, argmax, expected value, and matrix reading are parts of this concept. The best response is also the building block of equilibrium, one of the most basic concepts of game theory.

0.8.1 Best response

The answer to "what would I do if I knew what my opponent will do?" is the best response. The question contains an assumption: the opponent's choice is known and fixed. Under this assumption the game turns into a one-person decision problem and the answer is an argmax.

Definition 0.23

When the opponent plays s_2, player 1's **best response** is the set of strategies that maximize her own payoff:

$$BR_1(s_2) = \operatorname{argmax}_{s_1 \in S_1} u_1(s_1, s_2)$$

If the opponent plays a mixed y, the expected payoff is used in place of u_1; in matrix language the best pure response is the row of the largest element of Ay.

The definition has three parts. *The opponent's choice is fixed*: one column of the matrix is selected. *Only one's own payoffs are looked at*: the first numbers are read in that column. *An argmax is taken*: the row of the largest number is chosen. If there is a tie there is more than one best response, and all of them belong to the set.

Example 0.28

We find Ayşe's best responses in Guess the Card.

1. **Bora always says "Red."** The first numbers in the "Red" column: Red −3, Black 1. $BR_1(\text{"Red"}) = \text{Black}$ (Figure 0.28).
2. **Bora always says "Black."** The first numbers in the "Black" column: Red 1, Black −1. $BR_1(\text{"Black"}) = \text{Red}$.
3. **Bora mixes.** Suppose Bora says "Red" with probability $q = 1/2$. The Red row: $-3 \cdot 1/2 + 1 \cdot 1/2 = -1$. The Black row: $1 \cdot 1/2 + (-1) \cdot 1/2 = 0$. Since $0 > -1$, the best response is Black.
4. **Another mixture.** For $y = (1/4, 3/4)$ we found $Ay = (0, -1/2)$ in 0.7.3: the best response is Red.

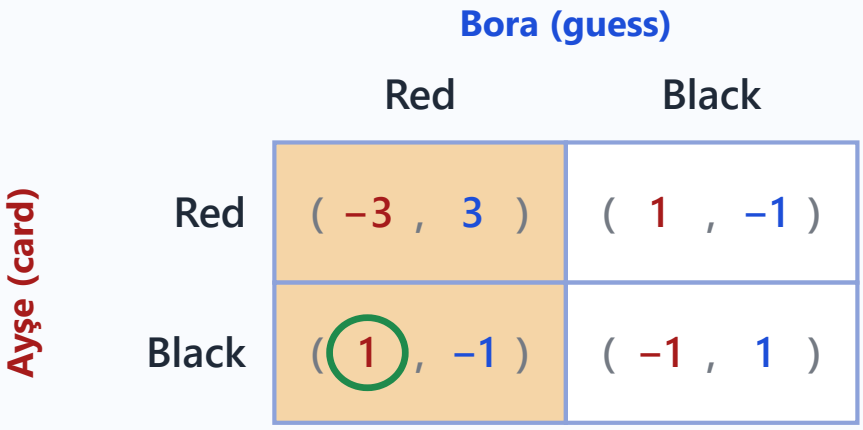


Figure 0.28: Ayşe's best response when Bora says "Red." Ayşe compares the first numbers in the "Red" column: −3 and 1. The circled 1 shows that Black is the best response.

The last two steps give an interesting observation: when Bora's mixture changed, Ayşe's best response changed too. At $q = 1/2$ Black is best, at $q = 1/4$ Red is best. Somewhere in between there must be a q at which the two moves bring the same. This value is found in 0.9.3 and is the key to how mixed strategies are computed.

Common mistake

Looking at the opponent's payoff (the second number) when choosing the best response. Ayşe compares only her own numbers; what Bora earns does not directly affect Ayşe's decision. Also, a best response is not fixed: it can change when the opponent changes. The sentence "Ayşe's best move is Black" is incomplete; the right form is "When Bora says 'Red,' Ayşe's best response is Black."

0.8.2 Trace: best responses in a 2×2 matrix

The best responses can be marked for the whole matrix at once. The method has two passes: first mark the row player's best response in each column, then the column player's best response in each row. In the game of Table 0.5, Ayşe chooses the row (Up/Down) and Bora the column (Left/Right).

Ayşe \ Bora	Left	Right
Up	(1, 2)	(3, 1)
Down	**(2, 3)**	(0, 0)

Table 0.5: The trace game. Each cell shows the (Ayşe, Bora) payoffs. In the green cell both players play their best responses.

Example 0.29

1. **If Bora plays Left,** Ayşe compares the first numbers in the Left column: Up 1, Down 2. *2 > 1* ⇒ Down. The 2 in the (Down, Left) cell is marked.
2. **If Bora plays Right:** Up 3, Down 0 ⇒ Up. The 3 in the (Up, Right) cell is marked.
3. **If Ayşe plays Up,** Bora compares the second numbers in the Up row: Left 2, Right 1 ⇒ Left. The 2 in (Up, Left) is marked.
4. **If Ayşe plays Down:** Left 3, Right 0 ⇒ Left. The 3 in (Down, Left) is marked. For Bora, Left is better in every case.
5. **Result.** The only cell in which both numbers are marked is (Down, Left) = (2, 3). In this cell both players play their best responses; nobody wants to change alone. The final state of the marking is shown in Figure 0.29.

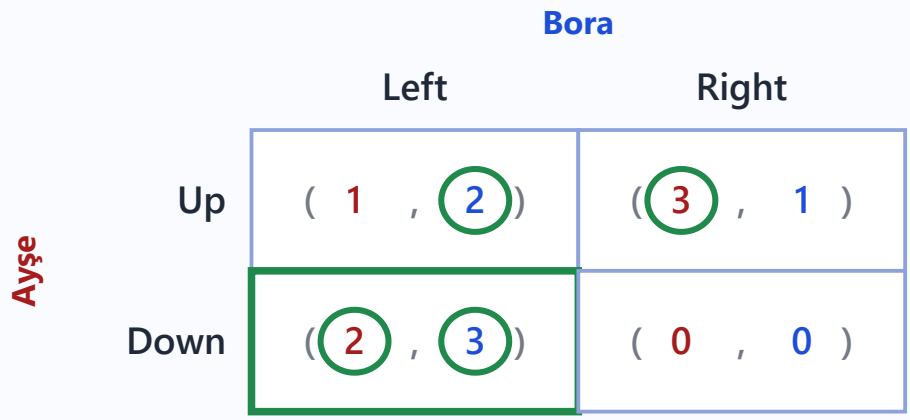


Figure 0.29: The final state of the marking. The circled numbers are best responses. The only cell in which both numbers are marked is (Down, Left) = (2, 3) (green frame).

The property found in the trace, that each player plays her best response to the other's choice, is called a **Nash equilibrium**. An equilibrium is a situation in which nobody wants to deviate alone. The marking method is the shortest way to find all the pure equilibria of a two-player matrix game: in each column mark the row player's largest number, in each row the column player's largest number; the cells with two marks are equilibria.

It is instructive to apply the method to Guess the Card. When Bora says "Red," Ayşe's best is Black; when Ayşe holds Black, Bora's best is to say "Black"; when Bora says "Black," Ayşe's best is Red; when Ayşe holds Red, Bora's best is "Red." The best responses form a cycle and no cell has two marks. Such games have no pure-strategy equilibrium; the players must mix. The probabilities of the mixture are found with the indifference equation.

Common mistake

Searching for the row player's best response along a row. The row player can change only her row; therefore her comparisons are made *within a column*, from top to bottom. The column player's comparisons are made *within a row*, from left to right. A second mistake is to count a cell with only one mark as an equilibrium: in (Up, Right) Ayşe's 3 is marked, but Bora wants to move to Left.

0.9 Equations and indifference

In game theory the unknowns are often probabilities: with what probability should a player play a move? These questions turn into equations. Most are linear (first degree), some are quadratic. The solution of these two kinds of equations prepares the ground for the most used equation in game theory, the indifference equation. The answers to the two questions in the introduction also rest on this equation.

0.9.1 Quadratic equations and the discriminant

An equation in which the square of the unknown appears is called a quadratic equation. Its graph is a parabola and the roots are the points where the parabola crosses the horizontal axis. A parabola may cross the axis at two points, touch the axis at just one point, or not meet the axis at all. A single number, the discriminant, tells which.

Definition 0.24

The **discriminant** of the equation $ax^2 + bx + c = 0$ ($a \neq 0$) is $\Delta = b^2 - 4ac$.

- If $\Delta > 0$, there are two distinct real roots,
- if $\Delta = 0$, a single (repeated) root,
- if $\Delta < 0$, no real root.

The roots are found with the formula:

$$x = (-b \pm \sqrt{\Delta}) / (2a)$$

The "±" sign in the formula abbreviates two separate computations: one with $+\sqrt{\Delta}$ and the other with $-\sqrt{\Delta}$. When $\Delta = 0$ the two computations give the same result; when $\Delta < 0$ the formula cannot be applied because the square root of a negative number is not a real number.

In game theory the unknown is often a probability. In that case, of the two roots of the equation only the one between 0 and 1 is meaningful. The other root is mathematically correct but cannot be a probability. This check must be made at the end of every solution.

Example 0.30

Suppose the equation $2p^2 + p - 1 = 0$ has been obtained for a probability p.

1. Read off the coefficients: $a = 2$, $b = 1$, $c = -1$.
2. Discriminant: $\Delta = 1^2 - 4 \cdot 2 \cdot (-1) = 1 + 8 = 9$. Since $\Delta > 0$ there are two roots; $\sqrt{\Delta} = 3$.
3. Roots: $p = (-1 \pm 3)/(2 \cdot 2) = (-1 \pm 3)/4$. With the plus sign $2/4 = 1/2$, with the minus sign $-4/4 = -1$.
4. A probability must lie between 0 and 1: $p = -1$ is discarded and $p = 1/2$ remains (Figure 0.30).
5. Check: $2 \cdot (1/2)^2 + 1/2 - 1 = 2 \cdot 1/4 + 1/2 - 1 = 1/2 + 1/2 - 1 = 0$ ✓.

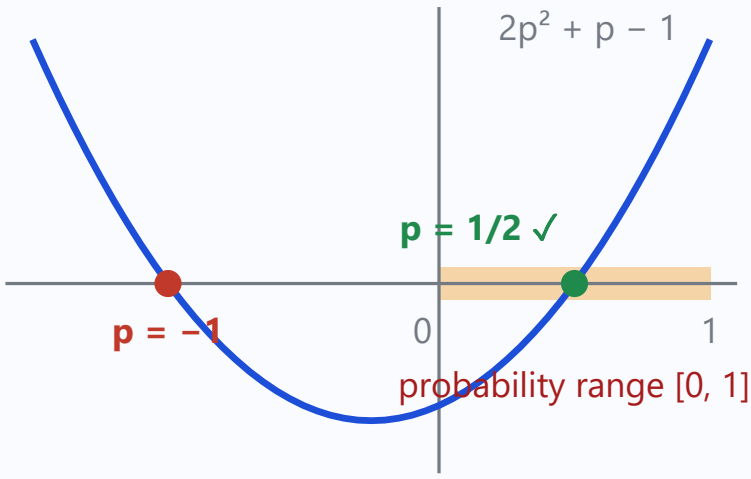


Figure 0.30: The parabola $2p^2 + p - 1$ crosses the axis at $p = -1$ and $p = 1/2$. The orange strip is the probability range [0, 1]; only $p = 1/2$ lies in it.

The other two cases of the discriminant can also be seen with examples. For $p^2 - p + 1/4 = 0$, $\Delta = 1 - 4 \cdot 1 \cdot 1/4 = 0$; the single root is $p = 1/(2 \cdot 1) = 1/2$, and the parabola merely touches the axis at this point. For $p^2 + p + 1 = 0$, $\Delta = 1 - 4 = -3 < 0$; the parabola never meets the axis and the equation has no real root.

Common mistake

Writing b instead of $-b$, or dividing only $\sqrt{\Delta}$ by $2a$. The fraction bar is under the whole of $(-b \pm \sqrt{\Delta})$. Missing the sign of $-4ac$ when c is negative is also common: in the example $-4 \cdot 2 \cdot (-1) = +8$; if the sign is missed, $\Delta = 1 - 8 = -7$ and one reaches a wrong conclusion such as "no root."

0.9.2 Linear equations in two unknowns

Two equations are needed to find two unknowns. A single equation gives infinitely many solution pairs; the second equation picks one of them. The idea of the solution is to eliminate one of the unknowns: either the two equations are subtracted from each other, or one unknown is solved from one equation and substituted into the other. The remaining equation in one unknown is solved and the value found is substituted back.

Example 0.31

In a game shop, 2 dice and 1 deck of cards cost 50 TL, and 1 die and 1 deck cost 35 TL. How much do a die and a deck cost each? Let the price of a die be z and the price of a deck d.

1. The equations: (1) $2z + d = 50$ and (2) $z + d = 35$.
2. Subtract (2) from (1): $(2z + d) - (z + d) = 50 - 35$. On the left $d - d = 0$ and $z = 15$ remains.
3. Substitute $z = 15$ into (2): $15 + d = 35 \Rightarrow d = 20$.
4. Check (1): $2 \cdot 15 + 20 = 50$ ✓. Check (2): $15 + d = 35$ with $d = 20$: $15 + 20 = 35$ ✓.

The typical form of these equations in game theory is this: for two probabilities x and y, one equation is "the probabilities sum to 1" and the other is a condition coming from the game. For example, if $x + y = 1$ and $x = 2y$, the second equation is substituted into the first (the substitution method): $2y + y = 1 \Rightarrow 3y = 1 \Rightarrow y = 1/3$ and $x = 2/3$. Check: $2/3 + 1/3 = 1$ ✓ and $2/3 = 2 \cdot 1/3$ ✓.

Example 0.32

Bora plays Left with probability q and Right with probability r. Ayşe's payoffs are 3 (Left) and 0 (Right) in Up, and 1 (Left) and 2 (Right) in Down. We want Ayşe's two rows to give equal expected payoffs.

1. The equations: $q + r = 1$ and $3q + 0 \cdot r = 1 \cdot q + 2r$.
2. The second equation simplifies: $3q - q = 2r \Rightarrow 2q = 2r \Rightarrow q = r$.
3. Substitute into the first: $q + q = 1 \Rightarrow q = 1/2$, $r = 1/2$.
4. Check: Up $3 \cdot 1/2 = 3/2$; Down $1 \cdot 1/2 + 2 \cdot 1/2 = 3/2$ ✓.

Common mistake

Forgetting one side or losing a sign when subtracting. In the operation $(2z + d) - (z + d)$, the minus in front of the parenthesis applies to both terms inside: $2z + d - z - d$. The right side must likewise be $50 - 35$. The values found are checked in both equations; checking in only one can hide an error in the other. In probability questions, forgetting to write the equation $r = 1 - q$ means trying to find two unknowns from a single equation.

0.9.3 The idea of indifference

What can be said about the moves that a player playing a mixed strategy mixes? The answer comes from a simple argument. If a player really mixes two of her moves, that is, gives both positive probability, the two moves must give her equal expected payoffs. If one were better, the player would always play it; giving probability to the worse one would lower her payoff for nothing.

Definition 0.25

If two actions of a player give equal expected payoffs against the opponent's strategy, the player is **indifferent** between these two actions. The **indifference principle**: if a player mixes several actions with positive probability and this is a best choice, all the actions she mixes give her equal expected payoffs.

The principle has a surprising consequence. A player's indifference depends on the *opponent's* probabilities, not on her own. The expected payoffs of Ayşe's two moves are computed from how often Bora plays what. The condition "Ayşe is indifferent" is therefore an equation about Bora's probability. Ayşe's own probability is found from Bora's indifference condition.

Example 0.33

In Guess the Card, let Bora say "Red" with probability q and "Black" with probability $1 - q$.

1. If Ayşe holds Red: $E[R] = -3 \cdot q + 1 \cdot (1 - q) = 1 - 4q$.
2. If Ayşe holds Black: $E[B] = 1 \cdot q + (-1) \cdot (1 - q) = 2q - 1$.
3. When $q < 1/3$, for example at $q = 0$, $E[R] = 1 > E[B] = -1$: Red is better.
4. When $q > 1/3$, for example at $q = 1$, $E[R] = -3 < E[B] = 1$: Black is better.
5. Only at $q = 1/3$ do the two lines cross and Ayşe is indifferent.

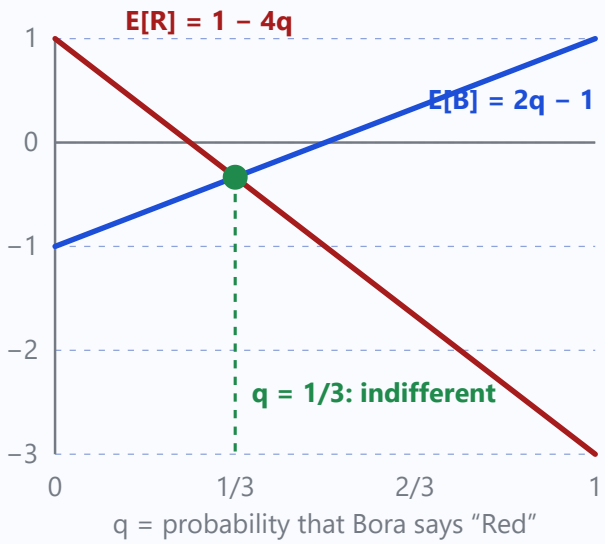


Figure 0.31: The expected payoff of Ayşe's two cards as a function of the probability q that Bora says "Red." The red line is $1 - 4q$ and the blue line is $2q - 1$; they cross at $q = 1/3$, where Ayşe is indifferent.

Figure 0.31 sums up the idea visually. Ayşe's best response for each q is the upper line: Red when q is small, Black when it is large. The crossing point of the two lines is the only value of q at which Ayşe can be willing to mix. The observation in 0.8.1 is explained here as well: $q = 1/4$ is to the left of the crossing and Red is best; $q = 1/2$ is to the right and Black is best.

Common mistake

Trying to find Ayşe's own probability from her own indifference equation. Ayşe's indifference determines Bora's value of q, because only q appears in $E[R]$ and $E[B]$. Ayşe's probability p of holding Red comes from the equation that makes Bora indifferent.

0.9.4 Trace: the indifference equation

The indifference equation is always set up and solved with the same four steps. In the trace below, p is the probability that Ayşe holds Red and q is the probability that Bora says "Red." The matrix used is given again in Figure 0.32.

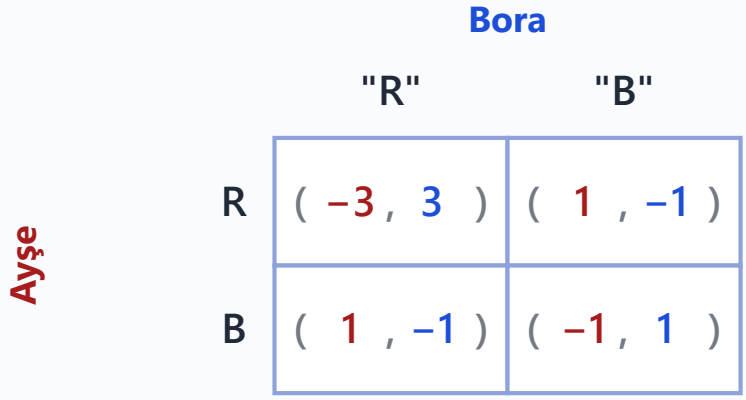


Figure 0.32: The matrix used in the trace. R and B are Ayşe's cards; "R" and "B" are Bora's guesses.

Example 0.34

1. **The unknown.** Bora says "Red" with probability q and "Black" with probability $1 - q$. The total is 1; no second unknown is needed.
2. **The expected payoff of each row.** Red: $E[R] = -3 \cdot q + 1 \cdot (1 - q) = 1 - 4q$. Black: $E[B] = 1 \cdot q + (-1) \cdot (1 - q) = 2q - 1$.
3. **Equate and solve.** $1 - 4q = 2q - 1 \Rightarrow 1 + 1 = 2q + 4q \Rightarrow 2 = 6q \Rightarrow q = 1/3$.
4. **Check.** $E[R] = 1 - 4/3 = -1/3$ and $E[B] = 2/3 - 1 = -1/3$ ✓. $q = 1/3$ lies between 0 and 1 ✓.

When Bora chooses $q = 1/3$, Ayşe gets the same expected payoff, $-1/3$, whichever card she holds.

Repeating the same four steps for Bora gives Ayşe's probability. Let Ayşe hold Red with probability p. Bora's payoffs are the negatives of Ayşe's. If Bora says "Red" he gets $3 \cdot p + (-1) \cdot (1 - p) = 4p - 1$, and if he says "Black" he gets $(-1) \cdot p + 1 \cdot (1 - p) = 1 - 2p$. Equating: $4p - 1 = 1 - 2p \Rightarrow 6p = 2 \Rightarrow p = 1/3$. Check: both sides are $1/3$. In this game the two players' probabilities happen to come out the same; in general they differ.

Let us put the results together. If Ayşe holds Red with $p = 1/3$ and Bora says "Red" with $q = 1/3$, each is indifferent against the other's strategy; neither can increase her payoff by changing her strategy. Ayşe gets $-1/3$ per round and Bora $+1/3$. The game favors Bora. It also makes sense that Ayşe holds Red with probability only 1/3: if Red is caught it costs her 3 chips, so she holds it rarely. This is the mixed-strategy equilibrium of the game that was seen to have no pure equilibrium in 0.8.2.

0.9.5 Back to the game: Rock-Paper-Scissors

The two questions in the introduction can now be answered. Both are computations of expected payoff and best response.

Is there a way of playing that cannot be beaten? Yes: *(1/3, 1/3, 1/3)*. Whatever the opponent plays, the expected payoff is 0. For example, if the opponent plays Paper: Rock gets −1, Paper 0, Scissors +1, and the expected payoff is $1/3 \cdot (-1) + 1/3 \cdot 0 + 1/3 \cdot (+1) = 0$. When the opponent plays Rock or Scissors the same computation also gives 0; if the opponent mixes, the result is the average of these zeros, again 0. From the point of view of indifference, the even mixture equalizes the expected payoff of the opponent's three moves. Nothing is left for the opponent to exploit.

What does someone who always plays Rock lose? The opponent's best response is always Paper, and she wins every round: -1 per round. A milder bias also loses. Against someone who plays *(1/2, 1/4, 1/4)*, the expected payoff of each of the opponent's moves is computed: Paper $1/2 \cdot 1 + 1/4 \cdot 0 + 1/4 \cdot (-1) = 1/4$, Rock $0 + 1/4 \cdot (-1) + 1/4 \cdot 1 = 0$, Scissors $1/2 \cdot (-1) + 1/4 \cdot 1 + 0 = -1/4$. The opponent plays Paper and the biased player gets $-1/4$ per round.

Player's mixture (R, P, S)	Opponent's best response	Player's expected payoff
(1, 0, 0): always Rock	Paper	−1
(1/2, 1/4, 1/4)	Paper	−1/4
(1/3, 1/3, 1/3)	all equal	0

Table 0.6: The opponent's best response to a mixture, and the expected payoff of the player who plays that mixture.

Table 0.6 shows a single principle: every deviation from the even mixture loses against an opponent who notices it. In general, against someone who plays (p_R, p_P, p_S), the opponent's Paper brings $p_R - p_S$, her Scissors $p_P - p_R$, and her Rock $p_S - p_P$. These three numbers sum to zero; so either all of them are zero (the even mixture) or at least one is positive and the opponent picks it.

How do people play? In laboratory experiments the frequency of each move comes out close to 1/3, yet successive moves are not independent: winners tend to repeat the same move and losers tend to change theirs (Wang, Xu, and Zhou, 2014). Although the long-run frequencies resemble the even mixture, the play becomes predictable because each round is not chosen independently of the previous one. An opponent who learns patterns exploits exactly this: she looks not at frequencies but at the question "what follows after the previous round?" The mixture *(1/3, 1/3, 1/3)* that the theory recommends is unbeatable only if it is carried out with an independent random choice in each round.

0.10 Game trees and growth rates

A matrix is for games in which the players choose simultaneously. In chess, in bargaining, or in a firm's decision whether to enter a market, the players move in turn and each later player sees the earlier player's move. Such games are drawn as a tree and solved with backward induction. Because trees grow quickly, the O and Θ notation for growth rates is also needed.

0.10.1 Reading a game tree

Definition 0.26

A **game tree** is used for **sequential games**:

- A **node** in a circle shows whose turn it is. The topmost node is called the **root**; the game starts there.
- The **branches** leaving a node are the actions that player can choose at that moment.
- The **leaves** at the bottom show the players' payoffs when the game ends; the numbers are written in order of moves.

In a tree, each path from the root to a leaf is one play of the game. The number of leaves is the number of possible outcomes of the game. The basic difference between a matrix and a tree is information: in a tree the later player knows which branch the earlier player chose and decides accordingly. That is why the same player can have more than one node in a tree; each node corresponds to a different history and a separate decision is made at each.

Example 0.35

Ayşe hides or shows her card. Bora, seeing what Ayşe did, plays on or folds (Figure 0.33).

1. The root belongs to Ayşe; it has two branches: Hide and Show.
2. At the end of each branch there is a node of Bora's. Bora has two nodes and decides separately at each: once after Hide, once after Show.
3. There are four leaves. Hide → Play: (1, 2); Hide → Fold: (3, 0); Show → Play: (0, 1); Show → Fold: (2, 3).
4. For example, the path Hide → Fold reaches the leaf (3, 0): Ayşe gets 3, Bora 0.

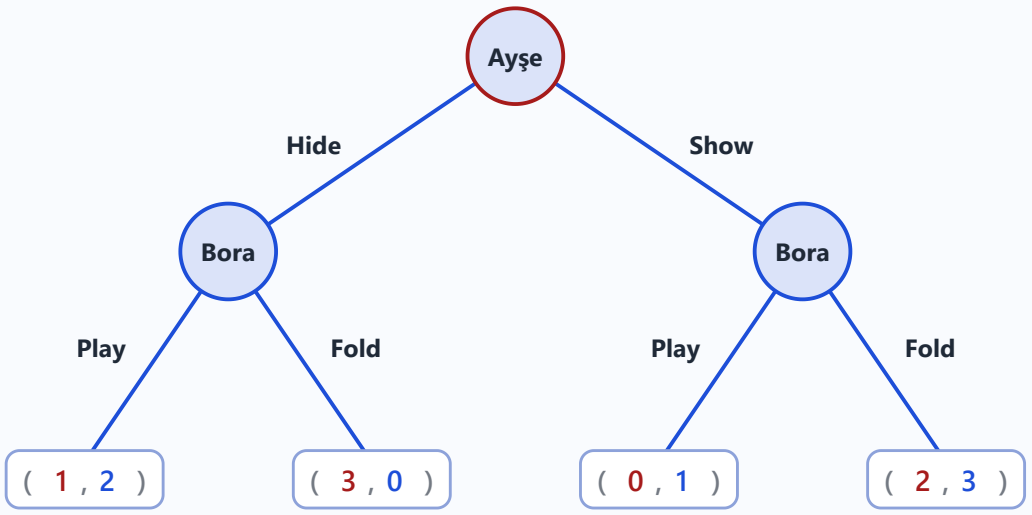


Figure 0.33: The tree of the card game. At the root Ayşe (red) chooses Hide or Show; at the two nodes Bora (blue) chooses Play or Fold. The leaves show the (Ayşe, Bora) payoffs.

Common mistake

Mixing up the order at a leaf: the first number is the payoff of the player who moves first (Ayşe) and the second is Bora's. Another is forgetting the difference from a matrix: here Bora sees Ayşe's move. The sentence "Bora plays" is incomplete; Bora may say different things after Hide and after Show.

0.10.2 Backward induction

To solve a sequential game, one starts not at the root but at the leaves. The idea is this: the decision of the player who makes the last move is easy, because nobody moves after her; she merely compares her own payoffs. Once her decision is known, the previous player can decide knowing what will happen at the end of each branch. This is repeated until the root is reached.

Definition 0.27

Backward induction: start with the nodes closest to the leaves. At each node, the player whose turn it is picks, from the payoff pairs at the ends of her branches, the one that is largest for *herself*; the node is replaced by that pair. The process ends when the root is reached. The pair found at the root is the outcome of the game; the choices at all the nodes together give the solution of the game.

Example 0.36

1. **Start: the last move.** Bora's two nodes are the nodes closest to the leaves.

2. **After Hide,** Bora compares the second numbers: Play 2, Fold 0 ⇒ Play. The value of this node becomes (1, 2).
3. **After Show,** Bora: Play 1, Fold 3 ⇒ Fold. The value of this node becomes (2, 3).
4. **Root.** Ayşe, knowing Bora's answers, compares the first numbers: Hide → 1, Show → 2 ⇒ Show.
5. **Result.** The path played is Show → Fold; the payoffs are (2, 3). The full solution: Ayşe plays Show; Bora plays Play after Hide and Fold after Show (Figure 0.34).

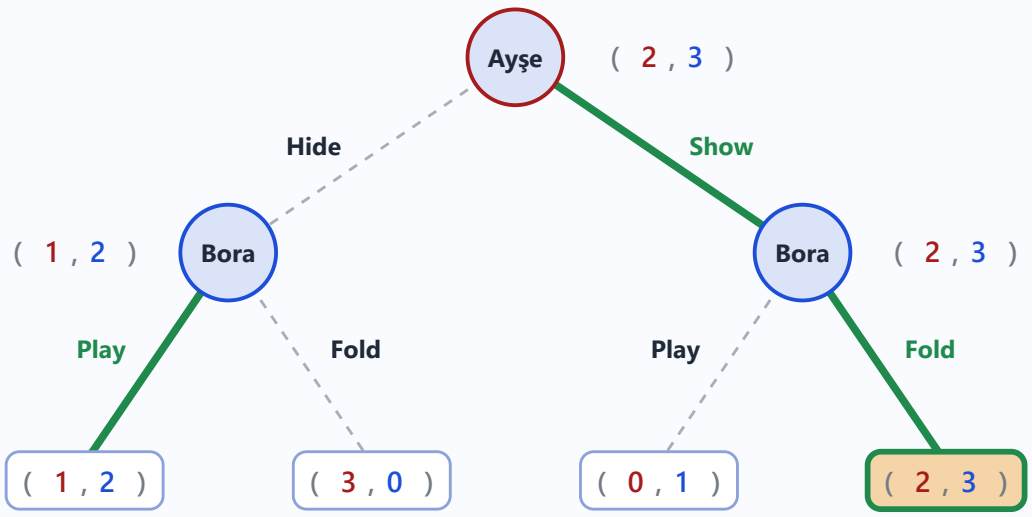


Figure 0.34: The result of backward induction. The green branches are the choice at each node; the dashed branches are those not chosen. The outcome path is Show → Fold, with payoffs (2, 3).

The result may be surprising at first: the leaf (3, 0) has the largest number for Ayşe, but the game never gets there. If Ayşe plays Hide, Bora chooses not Fold but Play, because Play gives him 2 and Fold 0. Ayşe computes this in advance and chooses Show, which gives her 2 instead of 1. This is the essence of backward induction: a player looks not at her own best leaf but at the best outcome that can be reached under the others' best responses.

In the full statement of the solution, Bora's choice at *every* node is given, including the node that is not played. The reason is that Ayşe's decision rests on Bora's reaction at the node that is not played: her reason for rejecting Hide is that Bora would play on after Hide. This is why, in sequential games, it matters that a strategy is a complete plan saying what to do in every situation.

Common mistake

Starting from the root: "Ayşe sees the largest number, 3, and says Hide." This reasoning assumes that Bora will help Ayşe. The right method moves from the leaves to the root: the last move is solved first. A second mistake is to look at the wrong player's number at each node: at

Bora's node the second numbers are compared, at Ayşe's node the first numbers.

0.10.3 Growth rates: O and Θ

Backward induction visits each node once. For small trees this is fast; but in the tree of a game like chess the number of nodes exceeds the number of atoms in the universe. To tell whether a method can be used in practice, we look at *how the number of steps grows* with the size of the input. The language for this is the O and Θ notation.

Definition 0.28

- $f(n) = O(g(n))$: for large n, f does not exceed a constant multiple of g. Formally, there exist $c > 0$ and n_0 such that $f(n) \le c \cdot g(n)$ for every $n \ge n_0$. This is an *upper bound*.
- $f(n) = \Theta(g(n))$: f grows at the same rate as g; for large n it stays between constant multiples of g from below and from above: $c_1 \cdot g(n) \le f(n) \le c_2 \cdot g(n)$.

Constant factors and slower-growing terms are dropped: $3n^2 + 5n = \Theta(n^2)$.

The last example can be checked against the definition. The lower bound is easy: $3n^2 + 5n \ge 3n^2$, so $c_1 = 3$. For the upper bound we use that for $n \ge 5$, $5n \le n \cdot n = n^2$: $3n^2 + 5n \le 3n^2 + n^2 = 4n^2$, so $c_2 = 4$ and $n_0 = 5$. Thus $3n^2 \le 3n^2 + 5n \le 4n^2$ and the function is $\Theta(n^2)$. The term $5n$ becomes insignificant next to n^2 for large n.

Example 0.37

1. **Ordering.** Common functions from slow to fast: $1 < \log\log n < \log n < n < n \log n < n^2 < 2^n$ (for large n).
2. **With numbers.** At $n = 16$, $\log_2 \log_2 n = 2$, $\log_2 n = 4$, $n^2 = 256$, $2^n = 65{,}536$. When n rises to 256, $\log_2 n$ is only 8, but 2^n rises to about 10^{77} (Table 0.7).
3. **A game tree.** If every node has 2 branches and the depth of the tree is d, the number of leaves is 2^d: the number of nodes doubles at each level. Since backward induction looks at each node once, it takes $\Theta(2^d)$ steps. At $d = 30$ there are about $2^{30} \approx 10^9$ leaves.

n	$\log_2 \log_2 n$	$\log_2 n$	n^2	2^n
16	2	4	256	65,536
256	3	8	65,536	$\approx 10^{77}$
65,536	4	16	$\approx 4.3 \cdot 10^9$	astronomically large

Table 0.7: A comparison of growth rates.

Table 0.7 shows why exponential growth is an obstacle. A computer doing 10^9 operations per second solves a binary tree of depth $d = 30$ in about a second; at $d = 60$ it needs about $2^{60}/10^9 \approx 1.15 \cdot 10^9$ seconds, more than 36 years. Doubling the depth does not double the time but multiplies it by a billion. That is why, in algorithmic game theory, problems that can be solved in "polynomial time" (such as n^2 or n^3) are considered easy, and problems thought to be solvable only in exponential time are considered hard.

Common mistake

Taking O to mean "exactly": $n = O(n^2)$ is true (it is only an upper bound), but $n = \Theta(n^2)$ is false, because n cannot be bounded from below by any constant multiple of n^2. O and Θ describe the trend at large n; at small n, a $\Theta(n^2)$ method can be faster than a $\Theta(n)$ method. For example, a method with n^2 steps is faster than a method with $100n$ steps for $n < 100$.

0.11 Common mistakes

Most of the mistakes in this chapter come from a few common sources: mixing up the order (in a profile, in a matrix, in a tree), confusing a number with an option (max and argmax), adding probabilities where one should not, and forgetting what a player can actually choose. Table 0.8 gives the most common mistake of each topic together with the correct version.

Topic	Mistake	Correct
Profile	(Rock, Paper) = (Paper, Rock)	Order matters: Ayşe first, then Bora
Counting	Counting order in an unordered choice ($4 \cdot 3 = 12$)	$C(4, 2) = 12/2! = 6$
Σ	$(\Sigma x_i)^2 = \Sigma x_i^2$	Compute the terms first, then add
Series	$1/(1 - \delta)$ for a sum starting tomorrow	$\delta/(1 - \delta)$; in which round does the first term come?
argmax	"argmax = 7"	argmax is an option (B); max is a value (7)
Derivative	Not checking the point $f' = 0$	$f'' < 0$ or neighboring values; endpoints
Exponents and ln	$2^2 + 2^3 = 2^5$; $\ln(a + b) = \ln a + \ln b$	Exponents add in a product; ln turns a product into a sum
Chain rule	$(e^{-x})' = e^{-x}$	Multiply by the inner derivative: $-e^{-x}$
Partial derivative	$\partial(qr)/\partial q = 0$	r is fixed but not zero: r
Probability	At least one 6: $1/6 + 1/6$	$1 - (5/6)^2 = 11/36$
Bayes	$P(A \mid B) = P(B \mid A)$	The prior counts too
Expected value	"E = 3.5 will come up"	The long-run average; may not occur on a single roll
U[0, 1]	Density = probability; $P(X = 0.5) > 0$	Probability is an area; that of a single point is 0
Mixed strategy	Taking turns	A single random pure move each round

Topic	Mistake	Correct
Softmax	$v_i / \Sigma v_j$	$e^{\lambda v_i} / \Sigma e^{\lambda v_j}$
Matrix	Writing b as the row player's payoff	(a, b): a is the row player's, b the column player's
Best response	Looking at the opponent's number	Fix the opponent's choice, compare one's own numbers
Quadratic	Dividing only $\sqrt{\Delta}$ by $2a$	$(-b \pm \sqrt{\Delta})/(2a)$; the probability is in [0, 1]
Indifference	Finding p from Ayşe's equation	p comes from the equation that makes Bora indifferent
Game tree	Starting from the root	From the leaves to the root: the last move first
O and Θ	$n = \Theta(n^2)$	$n = O(n^2)$; O is only an upper bound

Table 0.8: Common mistakes by topic.

0.12 Summary

- **Notation.** $S_1 \times S_2$ is the set of all strategy profiles; order matters in a profile. u_i assigns to a profile player i's payoff (a number). The number of profiles is counted with the product rule, unordered choices with $C(n, k)$.
- **Σ, series, max, and argmax.** With Σ, each term is computed first and then added. For $0 \le \delta < 1$, $1 + \delta + \delta^2 + \ldots = 1/(1 - \delta)$. max is a value; argmax is the option(s) attaining it; with a tie the argmax is a set.
- **Derivatives.** In a continuous choice the maximum is found with $f' = 0$ and checked with $f'' < 0$. For $u(q) = q(12 - q - r)$ the best quantity is $q^*(r) = \max\{0, (12 - r)/2\}$.
- **Exponents, e, and ln.** $x^a \cdot x^b = x^{a+b}$, $x^0 = 1$. ln is the inverse of e^x and turns a product into a sum; $(\ln x)' = 1/x$, $(e^x)' = e^x$.
- **Differentiation rules.** $(fg)' = f'g + fg'$; chain rule: outer derivative × inner derivative; $\partial u/\partial q$: the other variables are held fixed. Each player sets the partial derivative with respect to her own variable equal to zero; in the two-firm example $q = r = 4$.
- **Probability.** For equally likely outcomes, count and divide; multiply the probabilities of independent events, add those of events that cannot occur together; use the complement for "at least one." $P(A \mid B) = P(A \cap B)/P(B)$; Bayes: $P(A \mid B) = P(B \mid A) \cdot P(A)/P(B)$.
- **Expected value.** $E[X] = \Sigma\, x \cdot P(x)$: it is a long-run average, not the result of a single trial. A decision is made by the net expected value. For $X \ge 0$, $E[X] = \int P(X > t)\, dt$.
- **Continuous distributions.** In $U[0, 1]$ a probability is an area: $P(X \le x) = x$, a single point has probability 0, $E[X] = 1/2$. The integral is the area under a curve, the continuous version of Σ.
- **Mixed strategies.** A probability distribution over actions; a single pure move is drawn at random each round. Softmax: $P(i) \propto e^{\lambda v_i}$; equal at $\lambda = 0$, approaching argmax as λ grows.
- **The payoff matrix.** (a, b): a is the row player's payoff and b the column player's. The expected payoff is $x^T A y$; Ay is the expectation of each row.

- **Best response.** Fix the opponent's choice; take the argmax of one's own payoff. In a cell where both numbers are marked, both players play their best responses (Nash equilibrium).
- **Equations.** $x = (-b \pm \sqrt{\Delta})/(2a)$; the root that is a probability lies in [0, 1]. Indifference: the mixed actions give equal expected payoffs; one player's indifference determines the opponent's probability. In Guess the Card, $p = q = 1/3$ and Ayşe's payoff is $-1/3$.
- **Rock-Paper-Scissors.** $(1/3, 1/3, 1/3)$ is unbeatable (expected payoff 0); always Rock loses 1 per round, and $(1/2, 1/4, 1/4)$ loses 1/4 per round.
- **Backward induction.** From the leaves to the root; the player whose turn it is picks the largest of her own payoffs. In the card game the outcome is Show → Fold, (2, 3).
- **O and Θ.** $\log\log n < \log n < n < n \log n < n^2 < 2^n$; constants and small terms are dropped. In a binary tree, backward induction takes $\Theta(2^d)$ steps.

Table 0.9 matches each tool with the place where it is most used in game theory.

Tool	What it does	Where in game theory?
Cartesian product	Lists all profiles	The cells of a payoff matrix
Payoff function	Assigns a number to a profile	The statement of every definition
Geometric series	An infinite discounted sum	Repeated games
argmax / derivative	Finds the best option	Best response
Partial derivative	Slope while the other variables are fixed	Equilibrium in games with continuous strategies
Bayes' rule	Updates a belief after an observation	Games with incomplete information
Expected value	Long-run average	Payoff against mixed strategies
U[0, 1], integral, tail	Continuous uncertainty	Auctions
Softmax	Turns values into soft probabilities	Behavioral models, learning
Indifference equation	Finds the probabilities of a mixture	Mixed-strategy equilibrium
Backward induction	Solves a tree from leaves to root	Sequential games
O and Θ	Measures growth rate	The difficulty of computing an equilibrium

Table 0.9: The tools of the chapter and their uses in game theory.

0.13 Key terms

Cartesian product strategy profile payoff function product rule binomial coefficient C(n, k) geometric series discount factor argmax derivative chain rule partial derivative conditional probability Bayes' rule prior posterior expected value expected payoff uniform distribution probability density integral tail probability mixed strategy pure strategy support probability vector logit function softmax choice payoff matrix zero-sum game best response discriminant indifference principle sequential game game tree backward induction big-O and Θ notation

0.14 Test Yourself

1. Three players have 3, 3, and 2 strategies respectively. How many strategy profiles are there? If a guard can protect 3 of 5 targets, how many pure strategies does she have; if the targets were chosen "in order," how many ordered choices would be counted?

   Answer

   Product rule: $3 \cdot 3 \cdot 2 = 18$ profiles. Guard: $C(5, 3) = 5!/(3! \cdot 2!) = 120/12 = 10$ pure strategies. The ordered choices number $5 \cdot 4 \cdot 3 = 60$; since each triple is counted in $3! = 6$ different orders, $60/6 = 10$.

2. A cooperation yields 4 per round and future rounds are weighted by $\delta = 0.75$. What is the total discounted payoff starting from this round? What would it be if the payoff came only from the next round on?

   Answer

   Starting today: $4/(1 - 0.75) = 4/0.25 = 16$. Starting tomorrow: $4 \cdot 0.75/(1 - 0.75) = 3/0.25 = 12$. The difference is today's 4: $16 - 12 = 4$.

3. A bag holds two coins: one fair, the other a trick coin with heads on both sides. A coin is drawn at random from the bag and flipped, and it shows heads. What is the probability that the coin is the trick coin? If the same coin is flipped a second time and again shows heads, what does this probability become?

   Answer

   Prior $P(\text{trick}) = 1/2$. $P(\text{heads}) = 1/2 \cdot 1 + 1/2 \cdot 1/2 = 3/4$. Bayes: $P(\text{trick} \mid \text{heads}) = (1 \cdot 1/2)/(3/4) = 2/3$. For two heads, the fair coin's probability is $1/4$: $P(\text{two heads}) = 1/2 \cdot 1 + 1/2 \cdot 1/4 = 5/8$ and $P(\text{trick} \mid \text{two heads}) = (1/2)/(5/8) = 4/5$. By counting with 100 trials: all 50 trick-coin trials, and 12.5 of the 50 fair-coin trials, give two heads; $50/62.5 = 0.8$.

4. A wheel pays 0 TL with probability 1/2, 6 TL with probability 1/3, and 18 TL with probability 1/6; a spin costs 6 TL. What is the net expected gain? With the other prizes unchanged, what would the 18 TL prize have to be for the game to be fair?

   Answer

   The probabilities sum to $1/2 + 1/3 + 1/6 = 1$ ✓. $E = 0 \cdot 1/2 + 6 \cdot 1/3 + 18 \cdot 1/6 = 0 + 2 + 3 = 5$ TL; net $5 - 6 = -1$ TL. The fair prize y: $2 + y/6 = 6 \Rightarrow y/6 = 4 \Rightarrow y = 24$ TL.

5. In a zero-sum game the payoff matrix of the row player (Ayşe) is $A = [[2, -1], [-1, 1]]$ (rows Up, Down; columns Left, Right). If Bora plays $y = (1/2, 1/2)$, what is Ay and which is Ayşe's best response? With what q must Bora play Left and with what p must Ayşe play Up so that both players are indifferent? What is Ayşe's expected payoff?

   Answer

   $Ay = (2 \cdot 1/2 - 1 \cdot 1/2, -1 \cdot 1/2 + 1 \cdot 1/2) = (1/2, 0)$; the best response is Up. Ayşe's indifference: $2q - (1 - q) = -q + (1 - q) \Rightarrow 3q - 1 = 1 - 2q \Rightarrow q = 2/5$. Bora's payoffs are $-A$; his indifference: Left $-2p + (1 - p) = 1 - 3p$, Right $p - (1 - p) = 2p - 1 \Rightarrow 1 - 3p = 2p - 1 \Rightarrow p = 2/5$. Ayşe's payoff is $3 \cdot 2/5 - 1 = 1/5$ (check: $1 - 2 \cdot 2/5 = 1/5$ ✓).

6. The price is $20 - (q + r)$ and the cost is zero; the firm's profit is $u(q) = q(20 - q - r)$. If the opponent produces $r = 6$, what are the best quantity and the profit? In what two ways can it be shown that this point is a peak? What are the general best-response formula and $\partial u/\partial r$?

   Answer

   $u(q) = q(14 - q) = 14q - q^2$; $u'(q) = 14 - 2q = 0 \Rightarrow q = 7$, $u(7) = 49$. $u'' = -2 < 0$; neighbors $u(6) = u(8) = 48 < 49$. In general: $\partial u/\partial q = 20 - 2q - r = 0 \Rightarrow q^*(r) = \max\{0, (20 - r)/2\}$. $\partial u/\partial r = -q$.

7. Ayşe first says Left or Right; Bora sees it and responds. After Left, if Bora says a the payoffs are (2, 1), if he says b they are (0, 0); after Right, if he says c they are (3, 0), if he says d they are (1, 2) (Ayşe, Bora). What is the result of backward induction? How many leaves does a tree of depth d with 3 branches at every node have; how many for $d = 10$?

   Answer

   After Left, Bora chooses a ($1 > 0$) ⇒ (2, 1). After Right, Bora chooses d ($2 > 0$) ⇒ (1, 2). At the root Ayşe compares 2 and 1 and says Left: the outcome is (2, 1). (3, 0) is Ayşe's largest number but Bora does not choose c. The number of leaves is 3^d; for $d = 10$, $3^{10} = 59{,}049$. Backward induction takes $\Theta(3^d)$ steps.

8. The values of three moves are $v = (\ln 2, 0, 0)$ and $\lambda = 1$. What are the softmax probabilities? With two options, if $v_1 - v_2 = 1$ and $\lambda = \ln 4$, what is the logit probability $P(1)$? If 5 is added to all the values, do these probabilities change?

   Answer

   The exponentials are $e^{\ln 2} = 2$, $e^0 = 1$, $e^0 = 1$; the total is 4. The probabilities are $(2/4, 1/4, 1/4) = (1/2, 1/4, 1/4)$: the same as the biased two-coin mixture. Logit: $e^{-\lambda \cdot 1} = e^{-\ln 4} = 1/4$, $P(1) = 1/(1 + 1/4) = 4/5$. Adding 5 to the values multiplies every exponential by $e^{5\lambda}$; it cancels in the numerator and denominator and the probabilities do not change.

0.15 Further reading

- M. J. Osborne, *An Introduction to Game Theory* (2004), Chapter 17 (Mathematical appendix): a short worked summary of numbers, sets, functions, profiles, sequences, and probability, to the extent they are used in game theory.
- J. K. Blitzstein and J. Hwang, *Introduction to Probability* (2019), Chapters 1–5: presents counting, conditional probability, Bayes' rule, expected value, and the uniform distribution with many examples and simulations.
- E. Lehman, F. T. Leighton, and A. R. Meyer, *Mathematics for Computer Science* (2017): a free textbook for computer science students covering counting, sums, and asymptotic notation (O, Θ).
- Z. Wang, B. Xu, and H.-J. Zhou, "Social cycling and conditional responses in the Rock-Paper-Scissors game" (2014): a laboratory experiment showing that in Rock-Paper-Scissors people repeat after winning and switch after losing; short and readable.

0.16 References

1. J. K. Blitzstein, J. Hwang, *Introduction to Probability*, 2nd ed., CRC Press (Chapman & Hall), 2019.
2. T. H. Cormen, C. E. Leiserson, R. L. Rivest, C. Stein, *Introduction to Algorithms*, 4th ed., MIT Press, 2022 (Chapter 3: asymptotic notation).
3. E. Lehman, F. T. Leighton, A. R. Meyer, *Mathematics for Computer Science*, MIT, 2017.
4. R. D. McKelvey, T. R. Palfrey, "Quantal response equilibria for normal form games", *Games and Economic Behavior* 10(1), 6–38, 1995.
5. M. J. Osborne, *An Introduction to Game Theory*, Oxford University Press, 2004.
6. Z. Wang, B. Xu, H.-J. Zhou, "Social cycling and conditional responses in the Rock-Paper-Scissors game", *Scientific Reports* 4, 5830, 2014.

PART I · CHAPTER 1

Decision Theory

Game theory studies situations in which several decision makers affect one another. But first we must understand how a single decision maker chooses among options whose outcomes depend on chance. These notes present single-person decision theory: comparing risky options by expected utility, what utility numbers mean, attitudes toward risk, criteria to use when probabilities are unknown, updating beliefs with Bayes' rule as new information arrives, and computing the monetary value of information with decision trees. All of these tools are also building blocks of the strategic and collective decisions that follow.

LEARNING OUTCOMES

Expected utility and risk
Compute the expected value and the utility of a lottery; interpret risk attitudes through the certainty equivalent and the risk premium; apply the maximin, Laplace, and Savage criteria when the probabilities are unknown.

Utility scale
Explain the vNM axioms and the Allais and Ellsberg choices; show under which transformations utility numbers preserve the decision.

Bayesian decision
Update a prior with a test result; demonstrate base-rate neglect in a numerical example; choose the action with the smallest expected loss.

Decision trees
Find the best strategy by backward induction; compute the value of perfect information and of sample information.

1.1 Expected utility

Every day we make decisions whose outcomes we do not know in advance: whether to take an umbrella, which topic to study first for a course, whether to buy a product or wait. What these decisions have in common is that the outcome depends not only on our choice but also on something

we cannot control. Decision theory calls this "uncontrollable something" **nature** or **chance**. The basic tool for comparing options that depend on chance is expected utility.

1.1.1 Starting with a game: the Bayesian Decision Lab

Let us enter the subject through a small game. In the Bayesian Decision Lab an event is being planned, and there are two options: hold it *outdoors* or *indoors*. Whether the weather will be rainy or dry is hidden. The player knows only how likely rain is; this is called the **prior**. She also sees a noisy signal: a weather forecast. Then she makes her choice, the state is revealed, and she receives the payoff in the table.

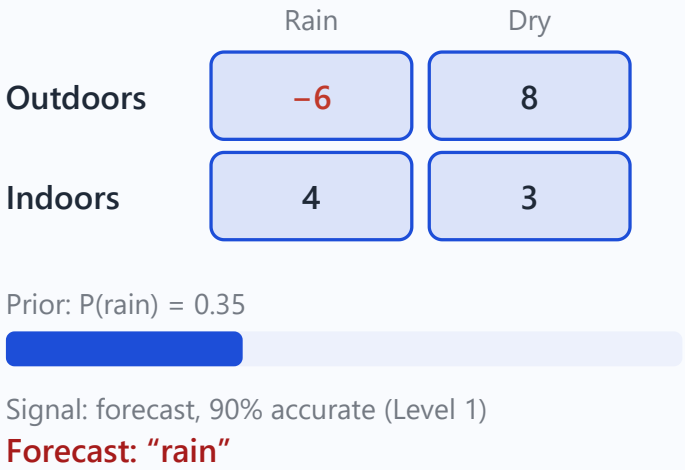


Figure 1.1: Level 1 of the Bayesian Decision Lab. Outdoors pays −6 in rain and 8 when dry; indoors pays 4 in rain and 3 when dry. The prior probability of rain is 0.35. The forecast, which is 90% accurate, has said "rain."

Figure 1.1 shows the first level of the game. The table is easy to read. Someone who chooses outdoors gets −6 if it rains and 8 if the weather stays dry. Someone who chooses indoors gets 4 in rain and 3 when dry. Outdoors offers a large gain but carries the risk of a large loss; indoors gives a reasonable result in both states.

The game begins with two questions. The first: if a 90% accurate forecast says "rain," what is the probability of rain? Most people's first answer is 90%. The correct answer is 82.9%, and the reason for the difference is the prior: rain is not very likely to begin with (0.35). How this number is found is computed step by step in 1.3.6. The second question is more philosophical: if a bad outcome occurs after a decision, was the decision bad? The answer may be "no." A good decision is the one that gives the best expectation with the information at hand. If chance goes badly, a good decision can yield a bad outcome. Decision theory judges decisions not by their outcomes but by the information available when they were made.

These two questions sum up the two main themes of the chapter. The first is how to update a belief with new information (Bayes' rule). The second is which option to choose once the belief is fixed (expected utility). The subject is built starting from the second.

1.1.2 Three kinds of decision making

Decision making is the process of choosing among options by evaluating their consequences. Decisions fall into three kinds according to what determines the outcome. Table 1.1 summarizes the distinction.

Kind	What determines the outcome	Tool	Example
Individual, under risk	Nature / chance (probabilities are known or estimated)	Expected utility, Bayes, decision tree	Investment, insurance, a medical test
Strategic	The choices of other rational actors	Best response, Nash equilibrium	Price competition, bargaining, sports
Collective	The joint choice and interaction of a group	Voting, common goods	Policy, corporate strategy

Table 1.1: Three kinds of decision making.

In an individual decision the decision maker chooses only against nature. Nature is not "a clever opponent": it does not rain in order to spoil the event. Nature's behavior is described with probabilities. In a strategic decision, what determines the outcome is other intelligent decision makers, who in turn think about what we will do. This situation of mutual thinking is the actual subject of game theory. In a collective decision a group decides together through a rule such as voting or sharing a common resource.

Why does this distinction matter? Because the analysis of strategic and collective decisions also uses the tools of single-person decision theory. A player who holds a belief about what her opponent will do makes her own choice with expected utility. If she does not know what "type" her opponent is, she updates her belief with Bayes' rule by observing his moves. A game in which moves are made in turn is solved with a game tree, which resembles a decision tree. That is why the tools in this chapter are the foundation of the rest of the course.

Decision theory also draws a distinction between risk and uncertainty. In **decision under risk** the probabilities are known or can be estimated reliably: rolling dice, roulette, insurance statistics. In **decision under uncertainty** the probabilities are unknown: whether a new product will succeed in the market, the success of a new technology. Most of the chapter is about decisions under risk; decisions under uncertainty are treated in 1.2.5 and 1.2.6.

1.1.3 Lotteries and expected value

The way to describe a risky option mathematically is to write it as a lottery. Here the word "lottery" does not mean only the game of chance in which a ticket is bought; it describes any option whose outcome depends on chance.

Definition 1.1

A **lottery** is a set of outcomes with probabilities. If the outcomes are $x_1, \ldots, x_k$ and their probabilities $p_1, \ldots, p_k$, the lottery is written $L = (p_1: x_1; \ldots; p_k: x_k)$. The probabilities are nonnegative and sum to 1: $p_1 + \ldots + p_k = 1$.

If the outcomes are numbers, the **expected value** of the lottery is defined as

$$E[X] = \Sigma\, p_i\, x_i = p_1 x_1 + p_2 x_2 + \ldots + p_k x_k$$

A sure amount is also a lottery with a single outcome: its probability is 1.

The Σ (sigma) sign means "add up." $\Sigma\, p_i\, x_i$ means multiplying each outcome by its probability and adding the products. The expected value is the average gain per play when the lottery is played many times. For example, if a game pays 10 TL on heads and nothing on tails is played 1000 times, heads comes up about 500 times; the total gain is about 5000 TL, an average of 5 TL per play. The expected value is likewise $\tfrac{1}{2} \cdot 10 + \tfrac{1}{2} \cdot 0 = 5$.

A two-option example is used often throughout the chapter:

- **Lottery A:** 900 TL with probability 50%, 0 TL with probability 50%.
- **Sure B:** 400 TL with probability 1.

Example 1.1

Let us compare the expected values of A and B.

1. A has two outcomes: 900 TL (probability ½) and 0 TL (probability ½). Each outcome is multiplied by its probability: $\tfrac{1}{2} \cdot 900 = 450$ and $\tfrac{1}{2} \cdot 0 = 0$.
2. The products are added: $E[A] = 450 + 0 = 450$ TL.
3. Since B is sure, its expected value is the amount itself: $E[B] = 1 \cdot 400 = 400$ TL.
4. Since $450 > 400$, a decision maker who looks only at the expected value chooses A.

Yet when people are asked, most choose B: they prefer a sure 400 TL to an even chance of 900 TL or nothing. Is this choice irrational? No; it can be explained by a consistent criterion.

Expected value is a powerful criterion that simplifies decisions, but it overlooks one thing: the value of money is not the same for everyone or at every amount. For someone with no money at all, the relief that 400 TL brings is great. The additional relief of going from 400 TL to 900 TL is not as great. This observation leads to the concept of utility.

Common mistake

- **Not summing the probabilities.** The probabilities of a lottery must sum to 1. In a lottery written as "900 TL with probability 50%, 0 TL with probability 40%," a 10% outcome has been forgotten.
- **Taking the expected value to be the "most likely outcome."** The expected value of A is 450 TL, but A never pays 450 TL: it pays either 900 or 0. The expected value is an average, not an outcome that occurs.
- **Leaving a percentage as a percentage in a product.** In a computation, 50% is written as 0.5 or ½; writing $50 \cdot 900$ inflates the result 100-fold.

1.1.4 Expected utility

Utility is a number showing how "good" an outcome is for the decision maker. A sum of money is an outcome; utility is the value of that outcome to the person. A **utility function** u assigns a number to each outcome. A commonly used example for money is $u(x) = \sqrt{x}$: the utility of 0 TL is 0, of 400 TL is 20, and of 900 TL is 30.

Definition 1.2

For a decision maker with utility function u, the **expected utility** of a lottery $L = (p_1: x_1; \ldots; p_k: x_k)$ is the number

$$E[u(L)] = \Sigma\, p_i\, u(x_i)$$

According to the **expected utility rule**, a rational decision maker chooses, among the options, the one with the largest expected utility; that is, she **maximizes** expected utility.

The definition is very similar to the definition of expected value. The only difference is that the probability multiplies the utility of the outcome ($u(x_i)$) rather than the outcome itself (x_i). The rule is also called "von Neumann–Morgenstern expected utility," after John von Neumann and Oskar Morgenstern, who established the theory systematically in 1944 (von Neumann and Morgenstern, 1944).



Figure 1.2: Lottery A and sure B. The red values show the option chosen by each criterion: expected value chooses A (450 > 400), expected utility ($\sqrt{x}$) chooses B (20 > 15).

Example 1.2

A decision maker with utility function $u(x) = \sqrt{x}$ chooses between A and B.

1. Find the utilities of A's outcomes: $u(900) = \sqrt{900} = 30$ and $u(0) = \sqrt{0} = 0$.
2. Multiply the utilities by the probabilities and add: $E[u(A)] = ½ \cdot 30 + ½ \cdot 0 = 15$.
3. B is sure: $u(B) = \sqrt{400} = 20$.
4. Since $20 > 15$, this decision maker chooses B.

Figure 1.2 shows the decisions of the two criteria side by side. For the same two options, expected value has chosen A and expected utility has chosen B. People who choose the sure 400 TL are not behaving irrationally; their utility functions are concave, that is, each additional lira of money adds a little less utility for them.

Expected monetary value and expected utility may not give the same decision. The difference is determined by the curvature of the utility function, that is, the decision maker's **risk attitude**. If the function is a straight line ($u(x) = x$), the two criteria give the same decision. If the function bends downward (like $\sqrt{x}$), the decision maker is risk averse. This topic is treated in detail in 1.2.

Why is the expected utility rule considered "rational"? There are two answers. The first is practical: when the same kind of decision is made many times, the rule recommends the choice that brings the best result on average. The utility function also reflects the decision maker's attitude toward risk. The second is theoretical and stronger: anyone whose preferences satisfy a few reasonable conditions behaves as if she were maximizing the expected value of a utility function. These conditions are described in 1.1.6.

1.1.5 The St. Petersburg paradox

The historical starting point of the idea of utility is a coin-flipping game discussed in the 18th century. A coin is flipped until heads comes up for the first time. If heads comes on the first flip the player gets 2 TL, if on the second 4 TL, on the third 8 TL; in general, if it comes on the k-th flip she gets 2^k TL. What is the most that should be paid to play this game?

Let us compute the expected value. The probability that heads first comes on the k-th flip is $(\tfrac{1}{2})^k$, because the previous $k-1$ flips must be tails and the k-th must be heads. The payoff is then 2^k. Each term is probability times payoff:

$$E[X] = \tfrac{1}{2} \cdot 2 + \tfrac{1}{4} \cdot 4 + \tfrac{1}{8} \cdot 8 + \ldots = 1 + 1 + 1 + \ldots$$

Each term is 1 and there are infinitely many terms. The sum goes to infinity: the expected value of the game is infinite. Someone who looks only at expected value should be willing to give her entire wealth to play this game. In reality, nobody pays more than a few tens of lira for it. This contradiction is called the **St. Petersburg paradox**.

Daniel Bernoulli answered this paradox in 1738 as follows: people evaluate not money itself but the utility of money, and as wealth grows the utility of each additional lira diminishes (Bernoulli, 1738). Gabriel Cramer had proposed a similar idea in 1728. Bernoulli used a logarithmic utility. The logarithm is a very slowly growing function: $\log_2 x$ says to what power 2 must be raised to give x ($\log_2 8 = 3$, $\log_2 1024 = 10$).

Example 1.3

Let the utility be $u(x) = \log_2 x$. If heads comes on the k-th flip, the payoff is 2^k and its utility is $\log_2 2^k = k$.

1. Expected utility: $E[u] = \tfrac{1}{2} \cdot 1 + \tfrac{1}{4} \cdot 2 + \tfrac{1}{8} \cdot 3 + 1/16 \cdot 4 + \ldots$
2. The first terms are 0.5, 0.5, 0.375, 0.25, 0.156, …, shrinking quickly. The sum does not go to infinity; it approaches 2.
3. The sure amount with utility 2 comes from the equation $\log_2 x = 2$: $x = 4$ TL.

This decision maker values the game at a sure 4 TL. Even though the expected value is infinite, the amount she is willing to pay is small. The paradox disappears with the concept of utility.

The St. Petersburg example teaches two things. First, very large payoffs with very small probabilities can inflate the expected value but hardly affect people's decisions. Second, diminishing marginal utility (each additional lira adding less utility) is the mathematical explanation of risk aversion.

1.1.6 The von Neumann–Morgenstern axioms

The expected utility rule is not an arbitrary formula. Von Neumann and Morgenstern showed that for every decision maker whose preferences satisfy four reasonable conditions there is a utility function, and that this person behaves as if maximizing expected utility. These conditions are called the **von Neumann–Morgenstern axioms**. An axiom is a basic assumption accepted without proof.

A notation is needed to state the axioms. $A \succ B$ means "A is preferred to B." $A \sim B$ means "the decision maker is indifferent between A and B"; both are equally good for her.

Definition 1.3

A preference relation over lotteries satisfies the following four axioms:

- **Completeness**: Any two options are comparable. For every A and B, either $A \succ B$, or $B \succ A$, or $A \sim B$.
- **Transitivity**: If $A \succ B$ and $B \succ C$, then $A \succ C$.
- **Continuity**: If $A \succ B \succ C$, the intermediate B can be matched with a mixture of the two ends: for some probability p, B $\sim$ (p: A; $1 - p$: C).
- **Independence**: Mixing the same common component with the same probability into two lotteries does not change the ranking. If $A \succ B$, then for every C and every $0 < p \leq 1$, (p: A; $1 - p$: C) $\succ$ (p: B; $1 - p$: C).

Theorem 1.1 (von Neumann–Morgenstern)

If a preference relation satisfies the four axioms, there is a utility function u such that, for any two lotteries, $A \succ B$ if and only if $E[u(A)] > E[u(B)]$. That is, the decision maker maximizes expected utility.

The proof of the theorem is beyond the scope of these notes. Still, it is easy to see why each axiom is reasonable and what happens when it is violated.

Completeness says the decision maker has an opinion in every case. A decision maker who says "I cannot compare them" cannot decide. If **transitivity** is violated, the decision maker can be exploited. Suppose someone prefers apple to pear, pear to cherry, and cherry to apple. Holding a cherry, she is told "give me a little money and I will swap your cherry for a pear"; she agrees. Then she swaps the pear for the apple, and the apple again for the cherry, each time for a little money. At the end of the cycle she again has the cherry but has less money. This cycle is called a *money pump*; transitivity is the protection against such exploitation.

Continuity means the following: let A be very good, C very bad, and B an outcome in between. For example A = 1000 TL, B = 500 TL, C = 0 TL. A lottery that gives A with high probability and C with low probability is better than B; a lottery that gives A with low probability is worse than B. Somewhere in

between there is a value of p at which the decision maker is indifferent between B and the lottery. This probability of indifference actually yields the utility function itself: if we choose $u(A) = 1$ and $u(C) = 0$, then $u(B) = p$. This is the justification of why utility numbers are "measurable."

Independence is the most debated axiom. What it says is this: if two options have a common component, the decision should not depend on that common component. Let a coin be flipped; if it shows heads, a choice will be made between A and B, and if tails, C will be received in either case. If it shows tails, the choice does not matter. If it shows heads, A is preferred. Then even before the coin is flipped, the option "A on heads" should be preferred. People's actual choices sometimes violate this axiom.

1.1.7 The Allais paradox

In 1953 the French economist Maurice Allais asked people to make two separate choices (Allais, 1953). The amounts are in millions of lira. Table 1.2 shows the four options.

Choice	Option	1 million	5 million	0
Choice 1	A	100%	—	—
	B	89%	10%	1%
Choice 2	C	11%	—	89%
	D	—	10%	90%

Table 1.2: Allais's two choices. Each cell gives the probability of that amount.

In the first choice most people pick A. A gives a sure 1 million; B can bring more, but with 1% probability it gives nothing. Giving up a sure million and taking on the risk of zero feels bad to people. In the second choice most people pick D. Here, in both options one most probably wins nothing; the difference between 10% and 11% looks small, while 5 million is much larger than 1 million. Each choice is understandable on its own. The problem is that together they contradict expected utility theory.

First let us see the relation between the options. A and B have a common component: both give 1 million with probability 89% (89% of A's 100%, and B's 89%). If this common 89% "1 million" component is replaced by 89% "0" in both options, C is obtained from A and D from B. A turns into 1 million with probability 11% and 0 with probability 89%; this is C. B turns into 5 million with probability 10% and 0 with probability 1% + 89% = 90%; this is D. By the independence axiom, replacing the common component should not change the ranking.

Example 1.4

For any utility function u, let us compute the two differences. For brevity, $u(1)$, $u(5)$, and $u(0)$ are the utilities of 1 million, 5 million, and 0, respectively.

1. $E[u(A)] = u(1)$, because A is sure.

2. $E[u(B)] = 0.89 \cdot u(1) + 0.10 \cdot u(5) + 0.01 \cdot u(0)$.
3. Difference: $E[u(A)] - E[u(B)] = (1 - 0.89) \cdot u(1) - 0.10 \cdot u(5) - 0.01 \cdot u(0) = 0.11 \cdot u(1) - 0.10 \cdot u(5) - 0.01 \cdot u(0)$.
4. $E[u(C)] = 0.11 \cdot u(1) + 0.89 \cdot u(0)$ and $E[u(D)] = 0.10 \cdot u(5) + 0.90 \cdot u(0)$.
5. Difference: $E[u(C)] - E[u(D)] = 0.11 \cdot u(1) - 0.10 \cdot u(5) + (0.89 - 0.90) \cdot u(0) = 0.11 \cdot u(1) - 0.10 \cdot u(5) - 0.01 \cdot u(0)$.

The two differences are exactly the same expression. If this expression is positive, both A $\succ$ B and C $\succ$ D must hold. If it is negative, both B $\succ$ A and D $\succ$ C must hold. The choice "A and D" would require the difference to be both positive and negative at once. This is impossible; that is, no utility function in the expected utility model explains the choice "A and D" (A $\succ$ B and D $\succ$ C).

The computation was done without choosing any utility function. Whether $u(x) = x$, $u(x) = \sqrt{x}$, or any other function is taken, the result does not change. For example, with $u(x) = x$ both differences are $0.11 - 0.50 = -0.39$; this person chooses B and D. The expected values of the four options are also different from one another: A is 1 million, B 1.39 million, C 0.11 million, D 0.5 million. So the choices are not "arbitrary because everything is equal" either.

Which axiom do Allais's choices violate? Not transitivity but **independence**. People give certainty more weight than probability calculus says they should. The difference between 100% and 99% feels much larger than the difference between 11% and 10%. This phenomenon is called the **certainty effect**. The certainty effect is one of the starting points of **prospect theory**, which tries to explain people's actual choices. Prospect theory uses probabilities not directly but after passing them through a weighting function; it inflates small probabilities and shrinks probabilities close to certainty.

The Allais paradox does not invalidate expected utility theory. The theory is used for two different purposes. *Descriptively*, it tries to describe how people decide; here deviations such as Allais's matter. *Normatively*, it says how a consistent decision maker ought to decide. For algorithms and artificial agents the second use is usually the important one: a software agent's preferences are required to be consistent.

1.1.8 What do utility numbers measure?

Numbers such as $u(400) = 20$ are used in expected utility computations. Do these numbers have a unit? Is it meaningful to say "20 units of happiness"? The answer is that the utility scale is like a thermometer. The Celsius and Fahrenheit scales use different zero points and different units; the freezing point of water is 0 on one and 32 on the other. But both scales order temperatures the same way, and one is obtained from the other by the formula $F = 1.8 \cdot C + 32$. Utility is like this: its zero point and unit are arbitrary.

Definition 1.4

A transformation of the form $v = a \cdot u + b$ ($a > 0$) is called a **positive affine transformation**. The transformation stretches the scale by a factor of a and shifts it by b.

Theorem 1.2 (Uniqueness of the utility scale)

If u is a decision maker's vNM utility function, every positive affine transformation $v = a \cdot u + b$ represents the same preferences: it changes no lottery decision. Conversely, every vNM utility function representing the same preferences is of this form; in this sense the utility scale is unique. Non-affine transformations, even increasing ones, can change lottery decisions.

The first half of the theorem is shown with a short computation. The expected utility of a lottery under v is

$$E[v(L)] = \Sigma\, p_i\,(a \cdot u(x_i) + b) = a \cdot \Sigma\, p_i\, u(x_i) + b \cdot \Sigma\, p_i = a \cdot E[u(L)] + b$$

The first equality is the definition. The second splits the sum in two and takes the constants outside. The third comes from the probabilities summing to 1. So the new expected utility of every lottery is the old one multiplied by a with b added. Since $a > 0$, the larger stays larger; the ranking does not change. If $a < 0$, the ranking would be reversed. That is why the condition is $a > 0$.

Utility	Lottery A	Sure B	Choice
$u = \sqrt{x}$	15	20	B
$v = 10\sqrt{x} + 5$	155	205	B
$w = (\sqrt{x})^2 = x$	450	400	A

Table 1.3: A (½: 900 TL; ½: 0 TL) and sure B (400 TL) under three different utility scales.

Example 1.5

The numbers in Table 1.3 are found as follows.

1. $v = 10\sqrt{x} + 5$ is an affine transformation ($a = 10$, $b = 5$). $v(900) = 305$, $v(0) = 5$; $E[v(A)] = ½ \cdot 305 + ½ \cdot 5 = 155$. By the formula as well: $10 \cdot 15 + 5 = 155$. For sure B, $v(400) = 10 \cdot 20 + 5 = 205$. $205 > 155$, so the choice is again B.
2. $w = (\sqrt{x})^2$ is also an increasing transformation: as $\sqrt{x}$ grows, its square grows too. But it is not affine. Since $w(x) = x$, $E[w(A)] = 450$ and $w(B) = 400$; the choice goes back to A.

Every increasing transformation preserves the ranking among sure outcomes (400 TL is always better than 0 TL). The ranking among lotteries is preserved only by affine transformations. That is why vNM utility is called **cardinal utility**: not only the order but also the ratios of differences are meaningful.

This result has a direct counterpart in game theory. Multiplying all the numbers in a player's payoff table by 2 and adding 5 changes none of that player's decisions. When studying a game, it is therefore permissible to convert payoffs into numbers that are easier to compute with. But a non-affine transformation, such as squaring the payoffs, can change risky choices.

Common mistake

- **"u(B) − u(A) = 5 units of happiness."** The size of the difference depends on the choice of scale: on the scale v the same difference is 50. What is meaningful is the ranking among the same person's options.
- **"Ayşe's utility is 20 and Bora's is 15; Ayşe is happier."** Each person's scale can be shifted and stretched arbitrarily and separately. vNM theory makes no comparisons of utility between persons.
- **"Every increasing transformation preserves the decision."** This holds for sure outcomes; for lotteries only positive affine transformations preserve the decision.

1.2 Risk attitudes

Different people react differently to the same lottery. Some prefer the sure amount, others try their luck. This difference is explained by the shape of the utility function. Risk attitude is measured with two numbers: the certainty equivalent and the risk premium. These two numbers also explain why insurance exists. When the probabilities are not known at all, other criteria are needed.

1.2.1 Concave utility and diminishing marginal utility

Let us look at the values of the function $u(x) = \sqrt{x}$: $u(0) = 0$, $u(100) = 10$, $u(400) = 20$, $u(900) = 30$. The first 100 TL raises utility by 10. Going from 100 to 400, that is, adding 300 TL more, again raises utility by only 10. Going from 400 to 900 takes 500 TL. The money needed for each additional 10 units of utility keeps growing. In other words, each additional lira adds less utility than the one before. This is called **diminishing marginal utility**. "Marginal" means "when one more unit is added."

The graph of a function with diminishing marginal utility bends downward; such a function is called **concave**. Concavity has a geometric definition: the line segment joining two points on the curve (the **chord**) lies below the curve. Conversely, a function that bends upward is called **convex**; its chord lies

above the curve. The chord of a straight line is the line itself.

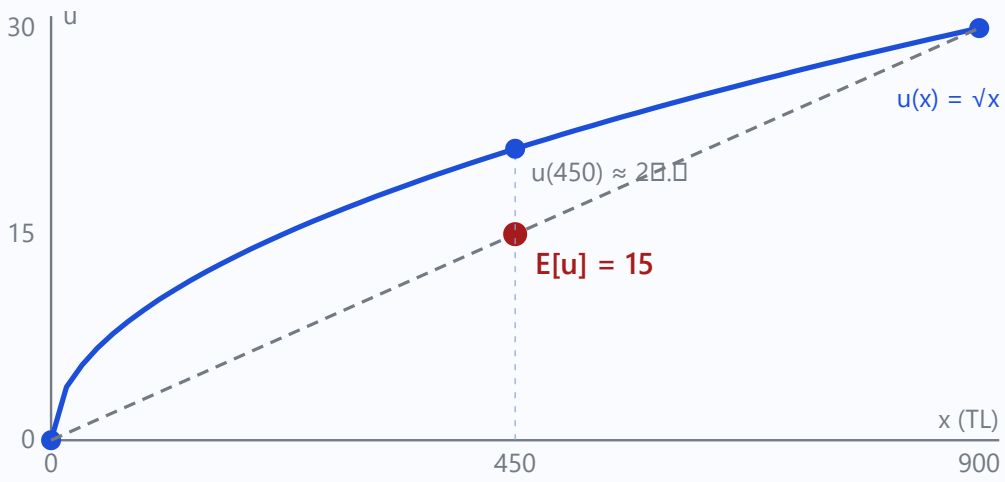


Figure 1.3: The concave utility $u(x) = \sqrt{x}$. The dashed line is the chord joining the two ends of the curve, the points (0, 0) and (900, 30). The expected utility of lottery A is the middle of the chord: $E[u] = 15$. If the same 450 TL were given for sure, its utility would be $u(450) \approx 21.2$, and the curve stays above the chord.

The chord lets us see the expected utility of a lottery. Lottery A gives one of the two ends with equal probability. Its expected utility is the average of the utilities of the two ends: $½ \cdot 0 + ½ \cdot 30 = 15$. This is the height of the point exactly in the middle of the chord, which falls at $x = 450$ on the horizontal axis, because the expected value is also the average of the two ends. At the same position in Figure 1.3 the height of the curve is $\sqrt{450} \approx 21.2$. The curve is above the chord: receiving 450 TL for sure gives more utility than a lottery whose average is 450 TL.

Theorem 1.3 (Jensen's inequality)

If u is concave, then $E[u(X)] \le u(E[X])$ for every lottery X. If u is convex, the inequality goes the other way: $E[u(X)] \ge u(E[X])$. If u is linear, the two sides are equal.

Jensen's inequality is the general form of the chord observation. The left side is the expected utility of the lottery (the point on the chord). The right side is the utility of receiving the expected value for sure (the point on the curve). For someone with concave utility, the lottery itself is worse than receiving its average for sure. This is the mathematical definition of risk aversion.

1.2.2 Certainty equivalent and risk premium

The way to express a risk attitude with a number is to ask: for what sure amount is this person willing to sell the lottery?

Definition 1.5

The **certainty equivalent** (CE) of a lottery X is the sure amount whose utility equals the lottery's expected utility:

$$u(CE) = E[u(X)], \text{ that is, } CE = u^{-1}(E[u(X)]).$$

The **risk premium** is the difference between the expected value and the certainty equivalent: *Risk premium* $= E[X] - CE$.

u^{-1} means undoing u: finding the amount whose utility is given. For $u(x) = \sqrt{x}$, the amount whose utility is 15 is the solution of the equation $\sqrt{x} = 15$. Squaring both sides: $x = 15^2 = 225$. So for $\sqrt{x}$, $u^{-1}(y) = y^2$.

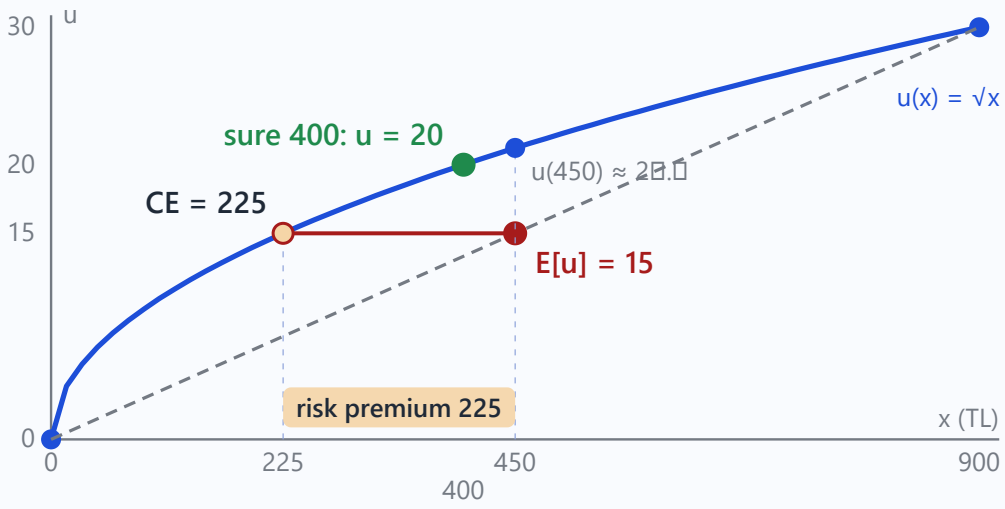


Figure 1.4: The certainty equivalent and the risk premium. Going horizontally from the point $E[u] = 15$ in the middle of the chord, one reaches the curve at $x = 225$: CE = 225 TL. The 225 TL between the expected value 450 and the CE is the risk premium. The utility of a sure 400 TL is 20, which is greater than 15.

Example 1.6

For lottery A (½: 900 TL; ½: 0 TL) and $u(x) = \sqrt{x}$, let us find the certainty equivalent and the risk premium step by step. Figure 1.4 shows each step.

1. **Utility of the ends.** $u(0) = 0$, $u(900) = 30$. These are the two end points of the curve.
2. **Expected utility.** $E[u] = ½ \cdot 0 + ½ \cdot 30 = 15$. This is the middle of the chord.
3. **Comparison.** The expected value is $E[X] = 450$ TL. The utility of a sure 450 TL is $\sqrt{450} \approx 21.2 > 15$. The person prefers receiving the expected value for sure to the lottery.

4. **Certainty equivalent.** The sure amount with utility 15: $\sqrt{CE} = 15 \Rightarrow CE = 225$ TL. This person is willing to sell the lottery for 225 TL.
5. **Risk premium.** $450 - 225 = 225$ TL. To avoid bearing the risk, the person is willing to give up 225 TL of the expected value.
6. **Decision.** The utility of a sure 400 TL is 20, which is greater than 15. This person chooses B. The same conclusion is reached another way: 400 TL is greater than CE = 225 TL.

The second route in the last step is an important shortcut. To compare a lottery with a sure amount, instead of comparing utilities it suffices to compare the CE with the sure amount. If the CE is larger than the sure amount, the lottery is chosen; if smaller, the sure amount. The shortcut works because u is increasing: a larger amount also has a larger utility.

Because the certainty equivalent is a sum of money, unlike utility numbers it can be compared across persons. It is meaningful to say "Ayşe would sell this lottery for 225 TL and Bora for 300 TL," and this shows that Bora is less risk averse. The CE is also independent of the choice of scale: even if $v = 10\sqrt{x} + 5$ is used, the sure amount whose utility is $E[v] = 155$ again comes out as 225 TL ($10\sqrt{x} + 5 = 155 \Rightarrow \sqrt{x} = 15$).

1.2.3 Three risk attitudes

The shape of the utility function defines three kinds of risk attitude. Table 1.4 compares the decisions of three utility functions for the same lottery A and sure B.

Attitude	Utility	E[u(A)]	u(400)	Choice	CE (TL)	Risk premium
Risk-averse	$\sqrt{x}$ (concave)	15	20	B	225	225
Risk-neutral	x (linear)	450	400	A	450	0
Risk-loving	$x^2/900$ (convex)	450	177.8	A	636.4	−186.4

Table 1.4: Three risk attitudes. Lottery A: 900 TL with probability 50%, 0 TL with probability 50% ($E[X] = 450$); B: a sure 400 TL.

Example 1.7

The second and third rows of Table 1.4 are computed as follows.

1. **Risk-neutral, $u(x) = x$.** Utility is the money itself. $E[u(A)] = \tfrac{1}{2} \cdot 900 + \tfrac{1}{2} \cdot 0 = 450$, $u(400) = 400$. $450 > 400$, so the choice is A. The amount whose utility is 450 is 450: CE = 450, risk premium $450 - 450 = 0$.
2. **Risk-loving, $u(x) = x^2/900$.** The ends: $u(0) = 0$, $u(900) = 900^2/900 = 900$. $E[u(A)] = \tfrac{1}{2} \cdot 0 + \tfrac{1}{2} \cdot 900 = 450$.
3. The utility of a sure 400: $400^2/900 = 160000/900 \approx 177.8$. $450 > 177.8$, so the choice is A.
4. CE: $x^2/900 = 450 \Rightarrow x^2 = 405000 \Rightarrow x = \sqrt{405000} \approx 636.4$ TL.

5. Risk premium: $450 - 636.4 = -186.4$ TL. The premium is negative.

What does a negative risk premium mean? A risk-loving person will not sell a lottery whose average is 450 TL for less than 636.4 TL. That is, she is willing to pay a price above the expected value to bear the risk. The three attitudes can be summarized as follows:

- **Concave → risk-averse.** $CE < E[X]$, premium > 0. She wants a reward for taking on risk. Most people behave this way, especially with large amounts.
- **Linear → risk-neutral.** $CE = E[X]$, premium = 0. She looks only at the expected value. Large firms behave approximately this way in decisions involving small amounts.
- **Convex → risk-loving.** $CE > E[X]$, premium < 0. She pays for risk. A person who buys a lottery ticket whose expected value is below the ticket price is an example.

These three cases follow directly from Jensen's inequality. For concave u, $E[u(X)] < u(E[X])$. The utility of the CE equals the left side and the utility of the expected value is the right side. Since u is increasing, the amount with the smaller utility is also smaller: $CE < E[X]$. For convex u everything goes the other way.

Common mistake

- **Confusing the CE with the expected utility.** $E[u] = 15$ is a utility number; CE = 225 is a sum of money. To find the CE one must convert the utility back into money (u^{-1}).
- **Writing the risk premium as $CE - E[X]$.** The definition is $E[X] - CE$. If written the other way, the premium of a risk-averse person comes out negative and the interpretation is reversed.
- **"A risk-loving person always chooses the lottery."** A risk-loving person chooses the lottery when the expected values are equal. If the expected value of the lottery is far below the sure amount, she too may choose the sure amount.

1.2.4 Why does insurance exist?

One of the most important applications of risk attitude is insurance. The insured pays a premium higher than her expected loss. The insurer profits from this premium. How is an agreement possible in which both sides come out ahead? The answer is that the two sides have different risk attitudes.

Example 1.8

Let a person's wealth be 900 (thousand TL). With probability 10% a fire breaks out and causes a loss of 800, leaving 100. The utility is $u(x) = \sqrt{x}$. What is the most she will pay for insurance that covers the entire loss?

1. **Expected utility without insurance.** With probability 90% wealth is 900, with probability 10% it is 100. $E[u] = 0.9 \cdot \sqrt{900} + 0.1 \cdot \sqrt{100} = 0.9 \cdot 30 + 0.1 \cdot 10 = 27 + 1 = 28$.
2. **Certainty equivalent.** $\sqrt{CE} = 28 \Rightarrow CE = 28^2 = 784$. She is willing to accept a sure 784 in place of a risky 900.
3. **The highest premium.** Insured wealth is $900 - premium$ and is certain. This amount must not be less than 784: $900 - premium \geq 784 \Rightarrow premium \leq 116$.
4. **Expected loss.** $0.1 \cdot 800 = 80$. Expected wealth is $900 - 80 = 820$.
5. **The insurer's lower bound.** The insurer sells a large number of independent policies. Some of the fires occur, but the total payout stays close to the sum of the expected losses. The insurer is therefore approximately risk-neutral, and every premium above 80 brings her an expected profit.

Result: every premium between 80 and 116 is profitable for both sides. For example, at a premium of 100 the insured's wealth is a sure 800; its utility is $\sqrt{800} \approx 28.3 > 28$. The insurer earns an average profit of 20 per policy.

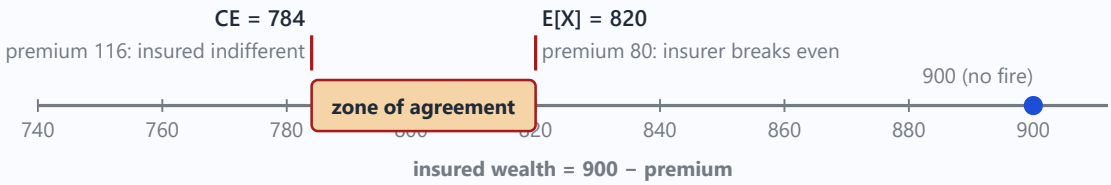


Figure 1.5: The zone of agreement for insurance. The horizontal axis is the insured's sure wealth (900 − premium). At a premium of 80 the wealth is 820 and the insurer breaks even. At a premium of 116 the wealth is 784 (the CE) and the insured is indifferent. Every premium in between benefits both sides.

Figure 1.5 shows this range on a number line. The width of the zone of agreement, $116 - 80 = 36$, is exactly the insured's risk premium: the difference between the expected wealth 820 and the certainty equivalent 784. The larger the risk premium, that is, the more the person avoids risk, the larger the gain that insurance can create.

The economic function of insurance is clear from this: risk is transferred to the party that can bear it most easily. For an individual a large loss can be a catastrophe; for the insurer it is one of thousands of policies. The same logic explains why large firms can behave as risk-neutral in small decisions: the outcome of a single decision is a small part of their total wealth.

1.2.5 The Ellsberg urn: ambiguity aversion

So far we have assumed that the probabilities are known. When rolling a die we know that each face comes up with probability 1/6. But nobody knows the probability of success of a new product. Under risk the probabilities are known; under uncertainty they are not. People do not behave the same way

in these two situations. Daniel Ellsberg showed this in 1961 with a simple experiment (Ellsberg, 1961).

An urn contains 90 balls. 30 of them are red. The remaining 60 balls are black or yellow, but how many are black is not known. Let s be the number of black balls in the urn; s can be any number from 0 to 60. A ball is drawn, and 100 TL is paid if the color that wins is drawn. Two bets are offered.

Bet	Option	Number of winning balls
1	red	30 (known)
	black	s (unknown)
2	red or yellow	$30 + (60 - s)$
	black or yellow	60 (known)

Table 1.5: The Ellsberg urn. s: the number of black balls in the urn (0–60, unknown).

The typical preferences are these: in the first bet most people choose "red," and in the second bet "black or yellow." What the two preferences have in common is that the probability of winning of the chosen option is known. In the first bet the probability of red is 30/90. In the second bet the probability of black or yellow is 60/90. The probabilities of the other options depend on s and are unknown.

Example 1.9

Is there a value of s that explains these two preferences? The prize is the same in both bets (100 TL), so the option with more winning balls should be preferred.

1. If the first preference is "red $\succ$ black," there must be more red balls than black: $30 > s$.
2. If the second preference is "black or yellow $\succ$ red or yellow," we must have $60 > 30 + (60 - s) = 90 - s$.
3. Rearrange the second inequality: $60 > 90 - s \Rightarrow s > 30$.
4. The first preference requires $s < 30$, the second $s > 30$. Both cannot hold at the same time.

No assignment of probabilities explains the two preferences together. The person's behavior cannot be explained by a consistent belief about the number of black balls in the urn.

This result cannot be rescued by the shape of the utility function either. In every option the prize is the same (100 TL or 0), so the curvature of utility merely scales two numbers; it does not change the perception of which option is more probable. The tendency to stay away from an option whose probability is unknown is called **ambiguity aversion**. It differs from risk aversion: risk aversion is the response to the distribution of outcomes with known probabilities, whereas ambiguity aversion is the response to the probabilities not being known.

The extreme form of ambiguity aversion is, when the probabilities are not known at all, to consider the worst case for each option and choose the option whose worst case is best. This rule is called **maximin**. In the Ellsberg urn maximin thinks as follows: in the bet on "black" the worst case is *s = 0* and the probability of winning is zero; the probability of winning on "red" is always 30/90. So red is chosen. In the second bet the worst case for "red or yellow" is *s = 60*, where the probability falls to 30/90; "black or yellow" always gives 60/90. Maximin explains both typical preferences.

1.2.6 When probabilities are unknown: criteria under uncertainty

An investor will choose one of three actions: a big investment, a small investment, or no investment at all. The result depends on the state of the economy: good, fair, or bad. The investor does not know the probabilities of these states. Table 1.6 shows the payoff of each action in each state. Rows are actions, columns are states.

Action \ state	Good	Fair	Bad
Big investment	200	50	−120
Small investment	90	40	−20
No investment	0	0	0

Table 1.6: An investment decision under uncertainty. Payoffs are in thousand TL; the probabilities of the states are unknown.

Since the probabilities are unknown, expected utility cannot be computed. Instead, criteria with different degrees of "pessimism" are used. All of these criteria look only at the numbers in the table. Such a problem is called a **decision problem under uncertainty**.

Definition 1.6

Let *U[a][d]* be the payoff of action *a* in state *d*. The four basic criteria are:

- **Maximin** (Wald): Find the worst outcome of each action; choose the action whose worst outcome is largest. $max_a\ min_d\ U[a][d]$.
- **Maximax**: Find the best outcome of each action; choose the one whose best outcome is largest. $max_a\ max_d\ U[a][d]$.
- **Laplace criterion**: All states are treated as equally likely; the action with the largest average is chosen.
- **Minimax regret** (Savage): In each state, the difference between the best attainable outcome and the outcome obtained is called the **regret**. Find the largest regret of each action; choose the action whose largest regret is smallest.

Behind each criterion is a point of view. Maximin is pessimistic: it acts on the assumption that "whatever I choose, nature will do the worst to me." Maximax is optimistic: it thinks "whatever I choose, the best will happen." Laplace says "if I know nothing, I should give every state equal probability"; it therefore represents ignorance with equal probabilities. Savage's criterion looks not at the outcome itself but at the regret one will feel afterwards (Savage, 1951).

Example 1.10

Let us apply the four criteria to Table 1.6.

1. **Maximin.** Row minima: Big −120, Small −20, None 0. The largest is 0. Decision: *No investment*.
2. **Maximax.** Row maxima: Big 200, Small 90, None 0. The largest is 200. Decision: *Big investment*.
3. **Laplace.** Row averages: Big *(200 + 50 − 120)/3 = 130/3 ≈ 43.3*; Small *(90 + 40 − 20)/3 = 110/3 ≈ 36.7*; None 0. The largest is 43.3. Decision: *Big investment*.
4. **Regret table.** First find the best attainable outcome in each state (column maxima): Good 200, Fair 50, Bad 0. Regret = best − obtained. Big: *(200 − 200, 50 − 50, 0 − (−120)) = (0, 0, 120)*. Small: *(110, 10, 20)*. None: *(200, 50, 0)*.
5. **Savage.** The largest regret per row: Big 120, Small 110, None 200. The smallest is 110. Decision: *Small investment*.

The same table has produced three different decisions: maximin "None," maximax and Laplace "Big," Savage "Small."

action \ state	Good	Fair	Bad	min	max	avg.	max regret
Big investment	200	50	−120	−120	200	4□.□	120
Small investment	90	40	−20	−20	90	3□.□	110
No investment	0	0	0	0	0	0	200
column best:	**200**	**50**	**0**				

Figure 1.6: The summary table of the uncertainty criteria. On the left the payoffs and the best of each column (200, 50, 0); in the middle the row minimum, maximum, and average; on the right the largest regret of each action.

Figure 1.6 collects all the intermediate results in one table. A common mistake in the regret computation is to compute the regret relative to the best of the row. Regret, however, is computed relative to the best of the column: "If I had known this state would occur, what would I have chosen and what did I lose?" In the bad state the best is not to invest at all (0). A person who invests big gets −120 in the bad state and feels a regret of 120.

A fifth criterion can be added to these four. The **Hurwicz criterion** stands between the two extremes. An **optimism coefficient** α (between 0 and 1) is chosen and each action is scored by

$$\alpha \cdot (\text{row max}) + (1 - \alpha) \cdot (\text{row min})$$

$\alpha = 0$ is maximin and $\alpha = 1$ is maximax. Scoring with $\alpha = 0.5$: Big $0.5 \cdot 200 + 0.5 \cdot (-120) = 40$, Small $0.5 \cdot 90 + 0.5 \cdot (-20) = 35$, None 0. The decision is the big investment.

Example 1.11

How does the decision of the Hurwicz criterion change as α changes?

1. Write the scores in terms of α: Big $200\alpha - 120(1 - \alpha) = 320\alpha - 120$; Small $90\alpha - 20(1 - \alpha) = 110\alpha - 20$; None 0.
2. Small is better than None when $110\alpha - 20 > 0 \Rightarrow \alpha > 2/11 \approx 0.18$.
3. Big is better than Small when $320\alpha - 120 > 110\alpha - 20 \Rightarrow 210\alpha > 100 \Rightarrow \alpha > 10/21 \approx 0.48$.

Result: for $\alpha < 0.18$ None is chosen, for $0.18 < \alpha < 0.48$ Small, and for $\alpha > 0.48$ the big investment. The more optimistic one is, the greater the risk taken.

Which criterion is correct? This question has no single answer. If the probabilities can be estimated, the best thing is to estimate them and return to expected utility. If they cannot, the choice of criterion reflects the decision maker's attitude. Where the risk of catastrophe is unacceptable (the design of a bridge, the treatment of a patient), maximin can be reasonable. For small decisions repeated often, Laplace or expected utility is more appropriate. In game theory maximin holds a special place: in two-person zero-sum games the opponent really is someone who wants to do the worst to you, and maximin becomes a natural criterion.

Common mistake

- **Computing regret within a row.** Regret is compared with the best outcome in the same *state* (column).
- **Choosing the largest regret in Savage.** First the largest regret of each row is found, then the *smallest* of these is chosen. That is where the name comes from: minimax regret.
- **Confusing maximin with minimax.** In a payoff table maximin is used (raising the worst payoff). In a cost or loss table the counterpart is minimax (lowering the largest loss).

1.2.7 Code trace: criteria()

The four criteria can be computed with a short Python function. The function takes the payoff table as a nested list: `U[i][j]` is the payoff of action `i` in state `j`. It returns the indices of the actions chosen by the four criteria.

```
def criteria(U):
    A, m = range(len(U)), len(U[0])
    mm = max(A, key=lambda i: min(U[i]))      # maximin
    mx = max(A, key=lambda i: max(U[i]))      # maximax
    lp = max(A, key=lambda i: sum(U[i]) / m)  # Laplace
    best = [max(r[j] for r in U) for j in range(m)]
    reg = [max(best[j] - r[j] for j in range(m))
           for r in U]            # max regret
    sv = reg.index(min(reg))                  # Savage
    return mm, mx, lp, sv
```

The code works line by line as follows:

- `A` is the range of action indices (0, 1, 2); `m` is the number of states (3).
- `max(A, key=…)` returns, among the indices in `A`, the one that makes the value of the `key` function largest. `lambda i: min(U[i])` is a small anonymous function: it takes the index `i` and gives the smallest value in that row. This line is a one-to-one counterpart of the maximin rule.
- `mx` builds the same structure with row maxima, and `lp` with row averages.
- The list `best` holds the largest value in each column: `r[j] for r in U` runs over the `j`-th element of all the rows.
- The list `reg` holds the largest regret of each row: in each column `best[j] - r[j]` is computed and the largest of these is taken.
- `sv` is the index of the smallest value in the list `reg`.

Step	Computed	Value
1	`A`, `m`	range(3), 3
2	row min. −120, −20, 0	`mm = 2` (None)
3	row max. 200, 90, 0	`mx = 0` (Big)
4	average 43.3, 36.7, 0.0	`lp = 0` (Big)
5	column max.	`best = [200, 50, 0]`
6	largest regret	`reg = [120, 110, 200]`
7	index of the smallest regret	`sv = 1` (Small)
8	return	`(2, 0, 0, 1)`

Table 1.7: Trace of `criteria(U)` with the payoffs of Table 1.6 (0: Big, 1: Small, 2: None).

The function returns `(2, 0, 0, 1)`. This quadruple is the same as the results found by hand: maximin None, maximax and Laplace Big, Savage Small. One detail is important: on a tie, `max` returns the first index it finds. If two actions had equal worst outcomes, the code would choose the one that comes first in the list. In a hand computation, too, one has to decide in advance how ties will be broken.

In the Laplace line the sum is divided by `m`. Since all rows are divided by the same number, the division does not change the ranking; the code would give the same decision by looking only at the sums. The division is there to make the result interpretable as an "average payoff." This is a small instance of the affine transformation result in 1.1.8: multiplying all the scores by a positive number preserves the ranking.

1.3 Bayesian decision

The expected utility computation uses probabilities. But how do the probabilities change when new information arrives? What is the probability of disease when a test comes out positive? What is the probability of rain when a weather forecast says "rain"? The rule for updating a belief with evidence is Bayes' rule. The updated probability goes directly into the expected utility computation.

1.3.1 Conditional probability

To understand Bayes' rule we first need conditional probability. *P(A)* is the probability of event A. *P(A | B)* means "the probability of A when B is known to have occurred" and is read "the probability of A given B." The right of the vertical bar is the known information, the left is the event asked about.

Definition 1.7

When *P(B) > 0*, the **conditional probability** is defined as

$$P(A \mid B) = P(A \text{ and } B) / P(B)$$

From this the product rule follows: $P(A \text{ and } B) = P(B \mid A) \cdot P(A)$.

The definition is easy to think about with a counting example. Suppose a class has 40 students; 10 wear glasses, and 4 of those who wear glasses are girls. Given that a randomly chosen student wears glasses, the probability that she is a girl is *4/10 = 0.4*. By the formula: *P(girl and glasses) = 4/40*, *P(glasses) = 10/40*; the ratio is *(4/40)/(10/40) = 4/10*. The condition narrows the population being counted: now we look only at those who wear glasses.

The most important point is this: *P(A | B)* and *P(B | A)* are generally different. The probability that a student with glasses is a girl is 0.4. The probability that a girl wears glasses depends on the number of girls in the class; if there are 20 girls it is *4/20 = 0.2*. The two probabilities divide the same "girl and

glasses" group by different populations. Confusing these two probabilities is the source of the most common mistake in Bayesian reasoning.

Three terms recur in the language of Bayesian decision:

- **Prior**: the belief before the evidence is seen. For example, the prevalence of a disease in the population, *P(sick)*.
- **Likelihood**: the probability of the observed evidence when the true state is known. For example, *P(+ | sick)*, the probability that a sick person's test comes out positive.
- **Posterior**: the updated belief after the evidence has been seen. For example, *P(sick | +)*.

1.3.2 Medical tests and natural frequencies

A disease has a prevalence of 1% in the population. There is a test for the disease. The test's **sensitivity** is 90%: the test is positive in 90% of the sick. The test's false-positive rate is 9%: the test is also positive in 9% of the healthy. A person's test came out positive. What is the probability that this person is really sick?

Most people, even most physicians, answer this question "about 90%." The correct answer is about 9%. The easiest way to see the large gap is to think in terms of numbers of people instead of probabilities. Converting probabilities into numbers of people in a population is called the method of **natural frequencies**. Gigerenzer and Hoffrage showed that many more people solve the problem correctly when the same problem is posed with natural frequencies (Gigerenzer and Hoffrage, 1995).

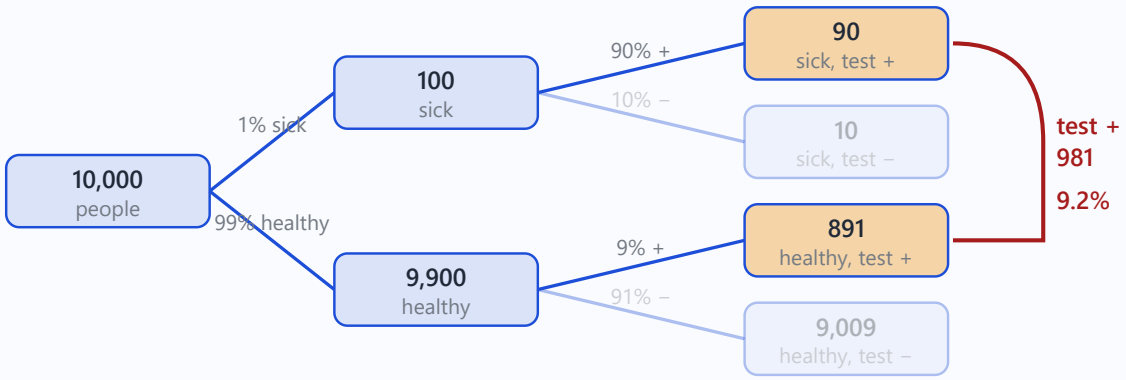


Figure 1.7: A medical test in a population of 10,000 people. 90 of the 100 sick and 891 of the 9,900 healthy test positive. Of the 981 who test positive, only 90 are sick: *P(sick | +) = 90/981 ≈ 9.2%*.

Example 1.12

Consider a population of 10,000 people. Figure 1.7 shows the numbers at each step.

1. **Prior.** The prevalence is 1%: *10,000 · 0.01 = 100* sick, *9,900* healthy.

2. **Test on the sick.** Sensitivity 90%: of the 100 sick, *100 · 0.9 = 90* test positive and 10 test negative.
3. **Test on the healthy.** False-positive rate 9%: of the 9,900 healthy, *9,900 · 0.09 = 891* also test positive. Because the healthy are so numerous, a small rate gives a large number.
4. **Everyone who tests positive.** *90 + 891 = 981* people.
5. **Posterior.** Of these 981 people only 90 are really sick: *P(sick | +) = 90/981 ≈ 0.092*, that is, about 9.2%.

Even though the test is 90% sensitive, a positive result is mostly a false alarm. Of every 11 people who test positive, about 10 are healthy.

This result does not mean the test is useless. Before the test the probability of disease was 1%; after the positive result it rose to 9.2%. The probability grew more than ninefold. But because the starting point was so low, the posterior is still below 10%. That is why in medicine a second, more definitive test is usually performed after a positive screening test.

1.3.3 Bayes' rule and base-rate neglect

The computation with natural frequencies is a special case of a general formula.

Theorem 1.4 (Bayes' rule)

Let H be a hypothesis (e.g., "sick") and E a piece of evidence (e.g., "test positive"). ¬H means that H does not hold ("healthy"). If *P(E) > 0*, then

$$P(H \mid E) = P(E \mid H) \cdot P(H) \,/\, [\, P(E \mid H) \cdot P(H) + P(E \mid \neg H) \cdot P(\neg H) \,]$$

The denominator is the total probability of the evidence, *P(E)*.

The formula follows from the definition of conditional probability in two steps. First step: by the definition, *P(H | E) = P(H and E) / P(E)*. By the product rule the numerator is written *P(E | H) · P(H)*. Second step: E can occur in two ways, either together with H or together with ¬H. Adding the probabilities of these two ways gives *P(E) = P(E | H) · P(H) + P(E | ¬H) · P(¬H)*. Putting the two steps together gives the formula.

In the medical test, *P(H) = 0.01*, *P(E | H) = 0.9*, *P(E | ¬H) = 0.09*. The numerator is *0.9 · 0.01 = 0.009*, the denominator *0.009 + 0.09 · 0.99 = 0.009 + 0.0891 = 0.0981*. The ratio is *0.009 / 0.0981 ≈ 0.0917*. This is the same as *90/981* in the natural frequencies; the only difference is that every number has been divided by 10,000. Natural frequencies are in fact Bayes' rule written with numbers of people.

Definition 1.8

Base-rate neglect is substituting the likelihood for the posterior without taking the prior (the base rate, the prevalence) into account. Its typical form is to take $P(+ \mid sick)$ to be $P(sick \mid +)$.

Base-rate neglect is a real problem. In a classic study, physicians and medical students were asked the following question: a test with a false-positive rate of 5% for a disease of prevalence 1/1000 has come out positive; what is the probability that the person is sick? Only 11 of 60 people reached the correct answer (about 2%). The most common answer was 95% (Casscells, Schoenberger, and Graboys, 1978).

Example 1.13

Let us solve the Casscells question with natural frequencies. The test's sensitivity is not given; let us assume that the test comes out positive for all the sick.

1. Take 100,000 people. Prevalence 1/1000: 100 sick, 99,900 healthy.
2. All 100 of the sick test positive.
3. 5% of the healthy test positive: $99{,}900 \cdot 0.05 = 4{,}995$.
4. Everyone who tests positive: $100 + 4{,}995 = 5{,}095$. Of these, 100 are sick: $100/5{,}095 \approx 0.0196$.

The correct answer is about 2%. The answer "95%" comes from taking the complement of the false-positive rate ($1 - 0.05$) to be the posterior.

The importance of the base rate is clear when the prevalence changes. Let the test in 1.3.2 (sensitivity 90%, false-positive 9%) now be used in a population where the prevalence is 10%. Of 10,000 people, 1,000 are sick and 900 of them test positive; of the 9,000 healthy, 810 test positive. $P(sick \mid +) = 900/1{,}710 = 10/19 \approx 52.6\%$. The same test gives a much more meaningful result in a population where the disease is more common. That is why screening tests are usually applied to people in a risk group.

Common mistake

- **Taking $P(E \mid H)$ to be $P(H \mid E)$.** "The test catches 90% of the sick" and "90% of those who test positive are sick" are different sentences.
- **Writing the denominator incompletely.** The denominator contains not only the true positives but also the false positives. If the false positives are forgotten, the posterior comes out as 1.

- **Computing the false positives from the whole population instead of from the healthy.** The 9% applies to the 9,900 healthy, not to the 10,000.

1.3.4 Code trace: bayes()

If one test result alone is not enough, a second test is done. For the second test the prior is no longer 1%; it is the posterior of the first test. Each new piece of evidence uses the previous posterior as its prior. This is called **sequential updating**. The function below does this with a loop. The tests are assumed to give results independent of one another once the person's true state is known.

```
def bayes(p, sens, fp, tests):
  for res in tests:
    if res == "+":
      num = sens * p
      den = num + fp * (1 - p)
    elif res == "-":
      num = (1 - sens) * p
      den = num + (1 - fp) * (1 - p)
    else:
      raise ValueError("result must be + or -")
    if den == 0:
      raise ValueError("impossible result")
    p = num / den
    print(res, round(p, 4))
  return p
```

The parameters of the function are: `p`, the initial prior; `sens`, *P(+ | sick)*; `fp`, the false-positive rate *P(+ | healthy)*; and `tests`, a list made of the results "+" and "-". For each result the loop does the following:

- If the result is "+", the likelihoods of a positive result are used: `sens` for the sick, `fp` for the healthy.
- If the result is "-", the likelihoods of a negative result are used: `1 - sens` for the sick, `1 - fp` for the healthy.
- `num` is the numerator of Bayes' rule; `den` is the total probability of the evidence.
- If any other sign arrives, or if `den` is 0 (the observation is impossible in both states according to the model), the function raises an error; in this case the posterior is undefined.
- The assignment `p = num / den` computes the posterior and writes it into the same variable. In the next round this value becomes the prior.

Round	res	prior p	num	den	new p
1	+	0.0100	*0.9 · 0.0100 = 0.0090*	*0.0090 + 0.09 · 0.9900 = 0.0981*	0.0917
2	+	0.0917	*0.9 · 0.0917 = 0.0826*	*0.0826 + 0.09 · 0.9083 = 0.1643*	0.5025

Round	res	prior p	num	den	new p
3	–	0.5025	*0.1 · 0.5025 = 0.0503*	*0.0503 + 0.91 · 0.4975 = 0.5030*	0.0999

Table 1.8: Trace of `bayes(0.01, 0.9, 0.09, ["+", "+", "-"])`.

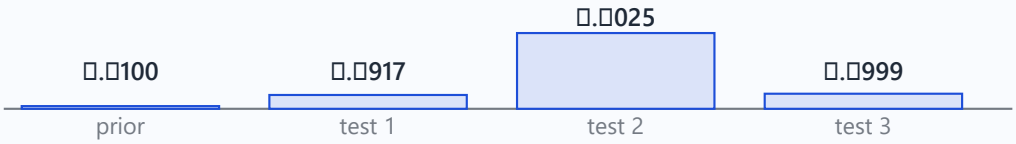


Figure 1.8: The probability of disease under sequential updating: prior 0.0100; after the first positive 0.0917; after the second positive 0.5025; when the third test comes out negative 0.0999.

The function prints the new probability in every round and finally returns `0.0999`. Figure 1.8 shows the course of the probability. The first positive raises the probability from 1% to 9.2%. The second positive (if the tests are independent given the disease status) carries it to about 50%. When the third test comes out negative, the probability falls back to 10%. A negative result is strong evidence, because the test comes out negative in only 10% of the sick, but in 91% of the healthy.

Sequential updating has two important properties. First, order does not matter: even if the same three results ("+", "-", "+") are processed in that order, the final probability is again 0.0999. Second, processing two positive results together in a single step gives the same result: the likelihood of two conditionally independent positives is $0.9 \cdot 0.9 = 0.81$ for the sick and $0.09 \cdot 0.09 = 0.0081$ for the healthy. The numerator is $0.81 \cdot 0.01 = 0.0081$, the denominator $0.0081 + 0.0081 \cdot 0.99 \approx 0.0161$; the ratio is again about 0.5025. But evaluating each test from scratch with the 1% prior is wrong: then the accumulated evidence of the two positives is lost.

1.3.5 The Bayesian decision rule: treat or wait?

Finding the posterior is half the job. The real question is what to do with this probability. After a positive test, should a physician treat the patient or wait? This decision depends both on the probability and on the cost of the two kinds of error. Treating a healthy person unnecessarily has a cost (side effects, expense). Not treating a sick person has a much larger cost.

Definition 1.9

The **Bayesian decision rule**: the prior is updated with the evidence; then the **expected loss** of each action is computed with the posterior. The action with the smallest expected loss is chosen.

Loss can be thought of as the opposite of utility: smaller is better. Working with losses is more natural in medicine and engineering, because the cost of errors is measured directly. Minimizing the expected loss is the same as treating the negative of the loss as utility and maximizing expected utility.

Example 1.14

Let the losses be: unnecessary treatment of a healthy person 30, failing to treat a sick person 200. The loss of a correct decision is 0. If the probability of disease is p:

1. **If treated,** a loss arises only if the person is healthy: $E[\text{loss} \mid \text{treat}] = (1 - p) \cdot 30$.
2. **If we wait,** a loss arises only if the person is sick: $E[\text{loss} \mid \text{wait}] = p \cdot 200$.
3. **Threshold.** Treatment is chosen if it has the smaller loss: $(1 - p) \cdot 30 < p \cdot 200 \Rightarrow 30 - 30p < 200p \Rightarrow 30 < 230p \Rightarrow p > 30/230 = 3/23 \approx 13.0\%$.
4. **After one positive,** $p \approx 0.0917$: treat $0.9083 \cdot 30 \approx 27.2$, wait $0.0917 \cdot 200 \approx 18.3$. Waiting has the smaller loss; the test is repeated.
5. **After two positives,** $p \approx 0.5025$: treat $0.4975 \cdot 30 \approx 14.9$, wait $0.5025 \cdot 200 \approx 100.5$. Treatment is chosen.

Case	p	E[loss | treat]	E[loss | wait]	Decision
1 positive	0.092	27.2	18.3	wait, retest
2 positives	0.503	14.9	100.5	treat

Table 1.9: The treatment decision. Losses: unnecessary treatment 30, untreated sick person 200; threshold $p > 3/23$.

The threshold formula can be written in general. If the loss of unnecessary treatment is c and the loss of an untreated sick person is k, treatment is chosen when $(1 - p)c < pk$, that is, when $p > c/(c + k)$. The interpretation of the formula is clear: the more expensive it is to miss the disease (the larger k), the lower the threshold. For a deadly disease even a low probability is enough for treatment. If the treatment itself is dangerous (c large), the threshold rises. The decision depends not only on the probability but also on the ratio of the losses.

Common mistake

- **"If the probability is below 50%, do not treat."** This rule is right only if the costs of the two errors are equal. Here the threshold is 13%.
- **Multiplying the losses by the wrong probability.** The loss of treatment is multiplied by the probability that the person is *healthy* ($1 - p$), because treatment does harm only when applied to a healthy person.

1.3.6 Back to the lab: the rain forecast

The first question of the game in 1.1.1 can now be answered. The prior probability of rain is 0.35 and the accuracy of the forecast is 90%. "Accuracy 90%" means this: if it is going to rain, the forecast says "rain" with probability 90%; if the weather is going to stay dry, it says "dry" with probability 90%. That is, $P(\text{"rain"} \mid \text{rain}) = 0.9$ and $P(\text{"rain"} \mid \text{dry}) = 0.1$.

Example 1.15

1. **Posterior.** $P(\text{rain} \mid \text{"rain"}) = 0.9 \cdot 0.35 / (0.9 \cdot 0.35 + 0.1 \cdot 0.65) = 0.315 / 0.38 \approx 0.829$. When the forecast says "rain," the probability of rain is not 90% but 82.9%.
2. **The expected utility** (EU) of the actions, when the probability of rain is p: outdoors $-6p + 8(1 - p) = 8 - 14p$; indoors $4p + 3(1 - p) = 3 + p$.
3. **Threshold.** Outdoors is better when $8 - 14p > 3 + p \Rightarrow 5 > 15p \Rightarrow p < 1/3$.
4. **With the prior alone** ($p = 0.35$): outdoors 3.10, indoors 3.35. Since $0.35 > 1/3$, indoors is chosen; but the difference is small.
5. **With the "rain" signal** ($p \approx 0.829$): outdoors −3.61, indoors 3.83. Indoors is chosen.
6. **With the "dry" signal**: $P(\text{rain} \mid \text{"dry"}) = 0.1 \cdot 0.35 / (0.1 \cdot 0.35 + 0.9 \cdot 0.65) = 0.035 / 0.62 \approx 0.056$. Outdoors 7.21, indoors 3.06. Outdoors is chosen.

Information	P(rain)	EU(outdoors)	EU(indoors)	Decision
prior only	0.35	3.10	3.35	indoors
"rain"	0.829	−3.61	3.83	indoors
"dry"	0.056	7.21	3.06	outdoors

Table 1.10: The Bayesian Decision Lab, level one (accuracy 90%).
EU: expected utility.

Table 1.10 shows why information is valuable. Without a signal, indoors is chosen. If the signal says "dry," the decision turns to outdoors and the expected utility rises from 3.35 to 7.21. Information is useful only where it can change the decision; the "rain" signal does not change the decision, but the "dry" signal does.

At the third level of the game the accuracy of the signal is 78% and the signal can be seen only by paying 1 unit. Is it worth buying? To answer this, the value of the signal is computed: the expected payoff when the decision is made after the signal, minus the expected payoff when the decision is made with the prior alone.

Example 1.16

1. **The probability of a "rain" signal.** *0.78 · 0.35 + 0.22 · 0.65 = 0.273 + 0.143 = 0.416*. The probability of a "dry" signal is *0.584*.
2. **If "rain" arrives.** *P(rain) = 0.273/0.416 ≈ 0.656*. Indoors is chosen: *3 + 0.656 ≈ 3.656*.
3. **If "dry" arrives.** *P(rain) = 0.22 · 0.35 / 0.584 = 0.077/0.584 ≈ 0.132*. Outdoors is chosen: *8 − 14 · 0.132 ≈ 6.154*.
4. **The expected payoff with the signal.** *0.416 · 3.656 + 0.584 · 6.154 ≈ 5.115*.
5. **The value of the signal.** The best payoff without the signal is 3.35. The value is *5.115 − 3.35 = 1.765*.

The information value of the signal is 1.765 and its price is 1. Since the value exceeds the price, the signal is worth buying.

The game has an "EU-optimal" counter. This counter counts not lucky outcomes but decisions that maximize expected utility given the information at hand. For example, someone who chooses outdoors after the "dry" signal is caught in rain with probability 5.6% and gets −6. That is a bad outcome, but the decision was good. Someone who in the same situation chooses indoors and by luck gets 4 has made a bad decision but received a good outcome. In the long run good decisions bring good results on average; a single outcome does not show the quality of the decision.

1.3.7 Updating beliefs about an opponent's type

Bayes' rule is a bridge from single-person decisions to games. In a game of incomplete information the unknown "state" is often the opponent himself: the cards in the opponent's hand, what kind of player the opponent is. This hidden characteristic of the opponent is called his **type**. The belief about the type is the prior; the opponent's move is the evidence.

Example 1.17

In a poker-like game the opponent is a bluffer with probability 20% and honest with probability 80%. The bluffer raises with probability 90% regardless of her hand. The honest player raises only when she has a strong hand; since a strong hand comes with probability 30%, the honest player raises with probability 30%. The opponent has raised. Let calling bring +30 against a bluffer and −30 against an honest player; let folding bring 0.

1. **Posterior.** *P(bluffer | raise) = 0.2 · 0.9 / (0.2 · 0.9 + 0.8 · 0.3) = 0.18 / 0.42 ≈ 0.429*.
2. **The expected value of calling.** *E[call] = 0.429 · 30 − 0.571 · 30 ≈ −4.3 < 0*. Folding is better.
3. **A second raise** (in another hand). The prior is now 0.429: *0.429 · 0.9 / (0.429 · 0.9 + 0.571 · 0.3) ≈ 0.692*.
4. *E[call] = 0.692 · 30 − 0.308 · 30 ≈ +11.5 > 0*. Now calling is profitable.

Observation	P(bluffer)	E[call]	Decision
prior only	0.200	−18.0	fold
1 raise	0.429	−4.3	fold
2 raises	0.692	+11.5	call

Table 1.11: The probability that the opponent is a bluffer, and the decision to call.

Since the gain and the loss are equal in size (30), calling is profitable only when $P(\text{bluffer}) > 0.5$. A single raise lifts the belief from 0.2 to 0.429 but does not bring it to the threshold. The second raise passes the threshold. The decision changes as evidence accumulates.

The common mistake in this example is to take $P(\text{raise} \mid \text{bluffer}) = 0.9$ to be $P(\text{bluffer} \mid \text{raise})$. This is the same as the base-rate neglect in the medical test. Because honest players are more numerous, most of the raises come from them: of the total raise probability 0.42, 0.24 comes from honest players, that is, about 57% of the raises belong to honest players.

This view explains behavior in card games. In poker and Hokm, bets and trump choices are evidence from which opponents update their beliefs. A bluff is an attempt to steer the opponent's updating: the bluffer pulls the opponent's posterior in the wrong direction by making the move an honest player would make. A smart opponent knows this and adjusts the likelihoods accordingly. This mutual thinking is exactly the point at which single-person decision theory passes into game theory.

1.4 Decision trees and the value of information

Real decisions are often not made in a single step. First some information is gathered, then a decision is made according to that information, then nature determines the outcome. The tool for organizing such sequential decisions is the decision tree. A product launch problem shows both how a tree is solved and how the monetary value of information is computed.

1.4.1 The elements of a decision tree

Definition 1.10

A **decision tree** consists of three kinds of elements:

- **Decision node** (□): The decision maker chooses. Its value is the largest value among its branches.
- **Chance node** (○): Nature chooses. Its value is the probability-weighted average of its branches.
- **Leaf**: The net outcome of a path.

A decision tree is a game tree in which nature also moves. The only difference is that nature does not think of its own payoff like a player; it acts only through probabilities. The tree is solved by **backward induction**: start from the leaves and work toward the root. The value of each node can be computed once the values of its children are known. At a chance node the average is taken, at a decision node the largest value.

Why backward? Because to make the right choice at a decision node, one needs to know the value of each branch. The value of a branch, in turn, depends on the decisions to be made later along that branch and on the chance events that will occur. Starting from the end keeps the needed information ready at every step. This method is also used on the trees of games such as chess.

1.4.2 The product launch problem

A firm will decide whether to launch a new product. If the market goes well (probability 2/5) it earns +500, if badly (probability 3/5) it loses 200 thousand TL. If the product is not launched the payoff is 0. Before deciding, the firm can commission a market survey for 20 thousand TL. The survey report is imperfect:

- If the market is really good, the report says "positive" with probability 80%: $P(\text{positive} \mid \text{good}) = 0.8$.
- If the market is bad, the report still says "positive" with probability 10%: $P(\text{positive} \mid \text{bad}) = 0.1$.

To solve the tree we need the probabilities at the chance nodes. If the survey is done, the first chance event is the report itself. After the report, the probability that the market is good must be updated according to the report. These are found with Bayes' rule.

Example 1.18

1. **The probability of a positive report.** $P(\text{positive}) = 0.4 \cdot 0.8 + 0.6 \cdot 0.1 = 0.32 + 0.06 = 0.38$. The probability of a negative report is 0.62.
2. **The market is good given a positive report.** $P(\text{good} \mid \text{positive}) = 0.32/0.38 = 16/19 \approx 0.842$.
3. **The market is good given a negative report.** In a good market the probability of a negative report is 0.2: $0.4 \cdot 0.2 = 0.08$. $P(\text{good} \mid \text{negative}) = 0.08/0.62 = 4/31 \approx 0.129$.

A positive report raises the probability of a good market from 0.4 to 0.842; a negative report lowers it to 0.129.

The net payoffs of the leaves must also be computed. On the paths where the survey is done, 20 thousand TL is deducted: launching in a good market gives $500 - 20 = 480$, launching in a bad market $-200 - 20 = -220$, and not launching $0 - 20 = -20$. On the paths without the survey the leaves are 500, −200, and 0.

1.4.3 Backward induction step by step

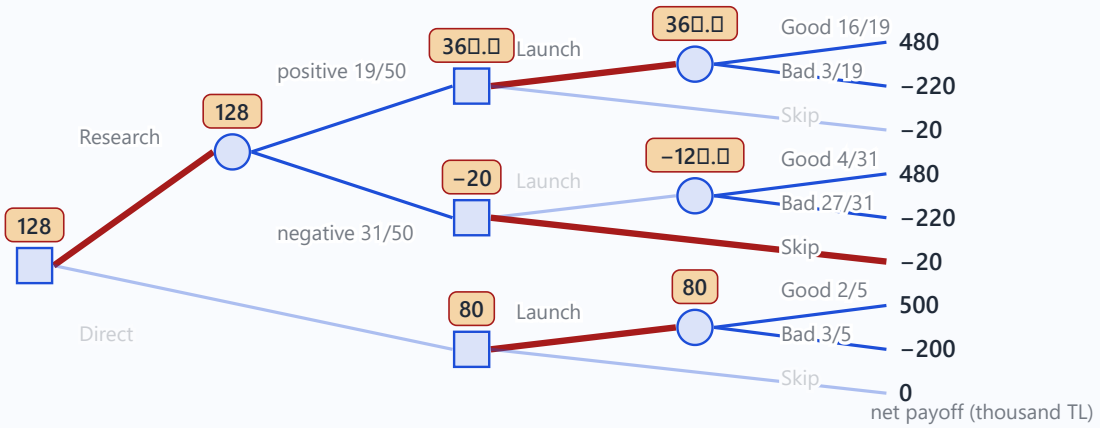


Figure 1.9: The product launch decision tree and the result of backward induction. The square is a decision node and the circle a chance node. The leaves show the net payoff (with 20 deducted if the survey is done). The thick red paths are the chosen decisions; the orange boxes are node values.

Example 1.19

Figure 1.9 shows the whole tree. Let us work from the leaves to the root.

1. **Launch without the survey (chance).** $2/5 \cdot 500 + 3/5 \cdot (-200) = 200 - 120 = 80$.
2. **The decision without the survey.** $\max(\text{Launch } 80, \text{Skip } 0) = 80 \Rightarrow$ Launch. The Skip branch is pruned.
3. **Launch after a positive report (chance).** $16/19 \cdot 480 + 3/19 \cdot (-220) = (7680 - 660)/19 = 7020/19 \approx 369.5$.
4. **The decision after a positive report.** $\max(369.5, -20) = 369.5 \Rightarrow$ Launch.
5. **Launch after a negative report (chance).** $4/31 \cdot 480 + 27/31 \cdot (-220) = (1920 - 5940)/31 = -4020/31 \approx -129.7$.
6. **The decision after a negative report.** $\max(-129.7, -20) = -20 \Rightarrow$ Skip. Only the cost of the survey is lost.
7. **The report (chance).** $0.38 \cdot 369.5 + 0.62 \cdot (-20) = 140.4 - 12.4 = 128$.
8. **The root (decision).** $\max(\text{Research } 128, \text{Direct } 80) = 128 \Rightarrow$ Research.

The best strategy: commission the survey; launch the product if the report is positive and do not launch if it is negative. The expected value of this strategy is 128 thousand TL.

The word "strategy" matters here. The result is not just "research"; it is a complete plan that says what to do against each possible report. In game theory, too, a strategy is defined this way: a complete plan that determines in advance what the player will do in every situation she may encounter.

The computation in the seventh step contains a small shortcut: $0.38 \cdot 7020/19 = 0.02 \cdot 7020 = 140.4$. Keeping the fractions to the end avoids rounding errors. Even if the rounded value 369.5 is used, the result comes out very close to 128 (140.41 − 12.4 = 128.01).

Common mistake

- **Reversing the rules.** Taking the average at a decision node and the maximum at a chance node is wrong. We cannot control nature, so at a chance node the average is taken. We do control our own decision, so we choose the best.
- **Letting nature choose in our favor.** Taking the largest branch at every node is overly optimistic: as if the market would always go well.
- **Following only the most probable branch.** Branches with low probability but large outcomes (like −220) must enter the average.
- **Deducting the survey cost twice.** The cost is either deducted from the leaves or at the very end from the root; if both are done, 20 is counted twice.

1.4.4 Code trace: value()

Backward induction is written briefly as a recursive (self-calling) function. In Python each node of the tree is represented by a tuple and each leaf by a number. A chance node has the form `("C", [(probability, child), …])` and a decision node `("D", [(name, child), …])`.

```
def value(n):
    if not isinstance(n, tuple):    # leaf
        return n
    kind, branches = n
    if kind == "C":                  # chance node
        return sum(p * value(c) for p, c in branches)
    return max(value(c) for name, c in branches)  # decision
```

The code works line by line as follows:

- If `n` is not a tuple it is a leaf; its value is the number itself and is returned directly. This is the stopping condition of the recursion.
- If it is a tuple, it is split into two parts: `kind` ("C" or "D") and the list `branches`.
- At a chance node the value of each branch is found recursively with `value(c)`, multiplied by its probability, and added.
- At a decision node the largest of the values of the branches is returned. The name of the branch (`name`) is not used in the computation; it is only for readability.

The launch tree is written in this form as follows:

```
tree = ("D", [
    ("Research", ("C", [
        (0.38, ("D", [("Launch", ("C", [(16/19, 480), (3/19, -220)])),
                      ("Skip", -20)])),
        (0.62, ("D", [("Launch", ("C", [(4/31, 480), (27/31, -220)])),
                      ("Skip", -20)]))])),
    ("Direct", ("D", [("Launch", ("C", [(0.4, 500), (0.6, -200)])),
                      ("Skip", 0)]))])
```

The call `value(tree)` traverses the tree depth-first: it enters a branch, descends to the deepest point of that branch, computes the value and returns, and then moves on to the next branch. Table 1.12 shows the order of the calls and the returned values. The "Stack" column shows the calls open at that moment; each new call is added on top of the stack and each return removes the top call.

Step	Call / return	Computation	Stack depth
1	`value(root)` — decision	children in turn	1
2	`value(Research)` — chance	children in turn	2
3	`value(positive)` — decision	children in turn	3
4	`value(positive·Launch)` — chance	$0.842 \cdot 480 + 0.158 \cdot (-220) = 369.5$	4
5	`positive` returns	$\max(369.5, -20) = 369.5$ → Launch	3
6	`value(negative·Launch)` — chance	$0.129 \cdot 480 + 0.871 \cdot (-220) = -129.7$	4
7	`negative` returns	$\max(-129.7, -20) = -20$ → Skip	3
8	`Research` returns	$0.38 \cdot 369.5 + 0.62 \cdot (-20) = 128.0$	2
9	`value(Direct·Launch)` — chance	$0.4 \cdot 500 + 0.6 \cdot (-200) = 80.0$	3
10	`Direct` returns	$\max(80, 0) = 80$ → Launch	2
11	`root` returns	$\max(128.0, 80.0) = 128.0$ → Research	1

Table 1.12: Trace of `value(tree)`. Leaf calls, and entering a node together with descending to its first child, are shown on a single line for brevity.

The function returns `128.0`, the same as the value found by hand. The code returns only the value, not which branch was chosen. To obtain the strategy as well, the name of the branch must be stored along with the `max` at a decision node. The running time of the function is proportional to the number of nodes in the tree, because each node is visited exactly once. This tree has 17 nodes: 4 decision nodes (root, positive, negative, Direct), 4 chance nodes, and 9 leaves.

1.4.5 The value of perfect information (EVPI)

Doing the survey raised the expected value from 80, the value without the survey, to 128. Two concepts are used to measure systematically how valuable information is. The first is the value of an information source with no defects at all.

Definition 1.11

The **expected value of perfect information** (EVPI) is the difference between the expected value we would obtain if we knew the state for certain in advance and the best expected value obtained without information:

> $$EVPI = E[\text{value with perfect information}] - E[\text{best value without information}]$$

It is easy to think of perfect information as an oracle. The oracle tells in advance whether the market will be good or bad. But what the oracle will say is not known in advance either: she says "good" with probability 40% and "bad" with probability 60%. So the value with perfect information is also an expected value.

Example 1.20

1. **No information.** The best action is to launch: $0.4 \cdot 500 + 0.6 \cdot (-200) = 80$.
2. **If the oracle says "good,"** the product is launched and 500 is earned. **If she says "bad,"** it is not launched and 0 is earned.
3. **The expected value with perfect information.** $0.4 \cdot 500 + 0.6 \cdot 0 = 200$.
4. **EVPI.** $200 - 80 = 120$.

No information source is worth paying more than 120 thousand TL for. An imperfect report cannot be more valuable than the oracle.

It is instructive to look at where the EVPI comes from. Without information the product is launched in every case. The oracle changes the decision only in the bad market: instead of launching she makes us choose not to launch and avoids the loss of −200. This gain occurs with probability 60%: $0.6 \cdot 200 = 120$. The EVPI is exactly the expected gain from the cases in which the decision changes.

1.4.6 The value of sample information (EVSI)

Real information sources are imperfect. A market survey is sometimes wrong. The value of imperfect information is smaller than the EVPI and must be computed separately.

Definition 1.12

The **expected value of sample information** (EVSI) is the difference between the expected value that would be obtained if an imperfect information source were free and the best expected value without information. An information source should be used only if its cost is less than the EVSI. We always have $0 \le EVSI \le EVPI$.

Example 1.21

1. **The value of the tree with the survey** is 128, but the cost of the survey (20) has already been deducted from this value.
2. **The value of a free survey.** Add the cost back: *128 + 20 = 148*. This is the expected value that would be obtained if the survey were free.
3. **EVSI.** *148 − 80 = 68*.
4. **Decision.** *EVSI = 68 > cost 20* ⇒ the survey should be done. The net gain is *68 − 20 = 48*; this also equals *128 − 80*.
5. **Efficiency.** *EVSI / EVPI = 68/120 ≈ 57%*. The imperfect report captures a little more than half of the value of perfect information.

Case	Computation	Value
No information (decide now)	Launch: *0.4 · 500 + 0.6 · (−200)*	80
Perfect information (oracle)	good → launch 500, bad → skip 0: *0.4 · 500 + 0.6 · 0*	200
EVPI	*200 − 80*	120
Sample information (survey, free)	128 from the tree + survey cost 20	148
EVSI	*148 − 80*	68

Table 1.13: The value of information, the product launch example (thousand TL).

The EVSI acts as a threshold. If the cost of the survey is below 68 the survey is done; if above, it is not. For example, if the survey cost 70 thousand TL, we would have *EVSI = 68 < 70*. The value of the survey path falls to *148 − 70 = 78*, which is less than 80, the value of launching directly. The best strategy would be to launch directly without a survey (expected 80).

Why is information valuable? Because it can change the decision. In this example a negative report leads to the decision "skip"; that is why the value of the survey is positive. If even a negative report still made launching look better, the same decision would be made whatever the report said, and the value of the survey would be zero. This principle is the general rule of gathering information: a test that cannot change any decision, however accurate it may be, is worth nothing.

Common mistake

- **Computing the EVSI without adding the cost back.** *128 − 80 = 48* is the net gain, not the EVSI. The EVSI is computed with the value of the free survey: *148 − 80 = 68*.
- **Deciding about the survey with the EVPI.** The EVPI is only an upper bound. If the cost is below the EVPI but above the EVSI, the survey is not done.
- **Taking the value without information to be zero.** The comparison is made not with "doing nothing" but with the best decision without information (here 80).

1.5 Discussion: better decisions under risk and uncertainty

The tools of the chapter together form a method of decision making. The method can be summed up in four principles.

- **The value of information is computed.** Up-to-date data improves the decision, but gathering information is not free. Gathering information pays only if the EVSI exceeds its cost. The EVPI is an upper bound on the highest price that could be paid.
- **Scenarios and the decision tree are written down explicitly.** Once the possible states and their probabilities are written, the tree can be solved backward. Drawing the tree makes overlooked options (e.g., "research first") visible.
- **The prior is taken into account.** Base rates enter the posterior through Bayes' rule. Thinking in natural frequencies ("out of 10,000 people, how many?") reduces base-rate neglect.
- **Opponents and diversification are considered.** If other decision makers, not nature, determine the outcome, game theory and simulation model the opponents' behavior. Spreading across mutually independent risks (portfolio diversification) reduces total risk; this is also why an insurer can behave as approximately risk-neutral.

All of these principles rest on one distinction: *the quality of the decision* and *the quality of the outcome* are different things. A decision that maximizes expected utility can produce a bad outcome if chance goes badly. Judging decisions by their outcomes means mistaking the share of chance for the share of the decision. A decision should be evaluated with the information available at the moment it was made.

1.5.1 A checklist for decision problems

When facing a decision problem, the following questions can be asked in order:

1. What are the options (actions)? What are the states that determine the outcome?
2. Are the probabilities of the states known? If so, the problem is a decision under risk (expected utility); if not, it is a decision under uncertainty (maximin, Laplace, Savage, Hurwicz).
3. If the outcomes are money, is the decision maker risk-neutral? If not, a utility function or the certainty equivalent is used.
4. Is there a piece of evidence (a test, a report, a signal)? If so, the prior is updated with Bayes' rule; the likelihood is not confused with the posterior.
5. Are the decisions sequential? If so, a decision tree is drawn and solved backward.
6. Can additional information be bought? The EVSI is computed and compared with the cost.

1.5.2 Four short problems

Four problems repeat the four main computations of the chapter. Each was obtained by changing the numbers of an example in the chapter.

Example 1.22

A lottery: 2,500 TL with probability 20%, otherwise 100 TL. Utility $u(x) = \sqrt{x}$. What are the expected value, the certainty equivalent, and the risk premium?

1. $E[X] = 0.2 \cdot 2{,}500 + 0.8 \cdot 100 = 500 + 80 = 580$ TL.
2. $E[u] = 0.2 \cdot \sqrt{2{,}500} + 0.8 \cdot \sqrt{100} = 0.2 \cdot 50 + 0.8 \cdot 10 = 10 + 8 = 18$.
3. $CE = 18^2 = 324$ TL.
4. Risk premium $= 580 - 324 = 256$ TL.

Example 1.23

If in the insurance example the probability of loss were 20% instead of 10%, what would the highest premium be?

1. $E[u] = 0.8 \cdot 30 + 0.2 \cdot 10 = 24 + 2 = 26$.
2. $CE = 26^2 = 676$.
3. The highest premium is $900 - 676 = 224$. The expected loss is $0.2 \cdot 800 = 160$.

The zone of agreement is between 160 and 224 and its width has grown to 64. As the probability of loss rises, the risk rises, and the risk premium grows.

Example 1.24

If the prevalence in the medical test were 10% (sensitivity 90%, false-positive 9%), what would $P(\text{sick} \mid +)$ be?

1. Numerator: $0.9 \cdot 0.1 = 0.09$. Denominator: $0.09 + 0.09 \cdot 0.9 = 0.09 + 0.081 = 0.171$.
2. $P = 0.09/0.171 = 10/19 \approx 52.6\%$.

When the base rate rises, a positive result becomes much more meaningful: 52.6% instead of 9.2%.

Example 1.25

If the survey in the decision tree cost 70 thousand TL, what would the best strategy be?

1. The EVSI is independent of the cost: 68.
2. $68 < 70$: the expected contribution of the survey does not cover its cost.
3. The best strategy is to launch directly; the expected value is 80.

1.5.3 Decisions in games

The concepts of the chapter find direct counterparts in many games: in Blackjack the decision to draw or stand rests on the probability of busting and on basic strategy; in Minesweeper the probability that each cell holds a mine is updated from the revealed numbers; in Pişti and Poker one reasons about the opponent's hidden cards from the cards played and the bets made; Hokm is a team card game of incomplete information in which trump choice and signaling matter; and in Backgammon, moves in a dice game are scored with a heuristic evaluation function.

All of these games share the same loop: a belief is held, updated with each new observation, and the move with the largest expected payoff under the current belief is chosen. In games with chance nodes such as Backgammon, choosing a move is a decision-tree search that averages over the possible outcomes of the dice.

1.6 Summary

- **Expected utility**: $E[u] = \Sigma\ p_i\ u(x_i)$. For lottery A (½: 900; ½: 0), $E[X] = 450$, but with $\sqrt{x}$, $E[u] = 15 < u(400) = 20$; expected value chooses A and expected utility chooses B. The St. Petersburg paradox shows that expected value alone is not enough.
- If the **vNM axioms** (completeness, transitivity, continuity, independence) hold, the decision maker maximizes expected utility. The Allais choices violate independence; the Ellsberg choices violate a consistent probability belief.
- **The utility scale**: the transformation $a \cdot u + b$ ($a > 0$) changes no lottery decision; increasing non-affine transformations can. The size of utility differences and interpersonal comparisons of utility are not meaningful.
- **Risk attitude**: concave → averse (CE 225, premium 225); linear → neutral (premium 0); convex → loving (negative premium). Jensen's inequality underlies this classification. In insurance, premiums between 80 and 116 benefit both sides.
- **Criteria under uncertainty**: on the same table, maximin chooses No investment, maximax and Laplace choose Big, and Savage chooses Small. The Hurwicz criterion moves between the two extremes with the optimism coefficient.
- **Bayesian decision**: prevalence 1%, sensitivity 90%, false-positive 9% ⇒ $P(\text{sick} \mid +) \approx 9.2\%$. In sequential tests the posterior is the prior of the next test. Treatment is chosen when its expected loss is smaller, that is, when $p > 3/23$.

- **Decision tree**: the average is taken at a chance node and the largest value at a decision node. Research (128) > no research (80); EVPI = 120, EVSI = 68. Information is valuable only if it can change the decision.

Concept	What it says	How it is computed
Expected utility	The value of risky options	$\Sigma\, p_i\, u(x_i)$
Certainty equivalent	The sure amount equivalent to a lottery	$u^{-1}(E[u])$
Risk premium	The price of getting rid of risk	$E[X] - CE$
Maximin / Savage	Decision without probabilities	Raise the worst case / lower the largest regret
Bayes' rule	Updating a belief with evidence	Numerator / (numerator + false alarms)
Bayesian decision rule	The best action under the posterior	Smallest expected loss
Backward induction	The best strategy in sequential decisions	Average at chance, maximum at decision
EVPI / EVSI	The monetary value of information	Value with information − value without

Table 1.14: The concepts of the chapter at a glance.

1.7 Key terms

decision making decision under risk decision under uncertainty lottery utility function
expected utility St. Petersburg paradox von Neumann–Morgenstern axioms
completeness / continuity / independence axioms Allais paradox certainty effect
affine transformation cardinal utility diminishing marginal utility concave convex
Jensen's inequality risk attitude risk-averse risk-neutral risk-loving certainty equivalent (CE)
risk premium Ellsberg paradox ambiguity aversion maximin maximax Laplace criterion
minimax regret Hurwicz criterion coefficient of optimism conditional probability prior likelihood
posterior Bayes' rule sensitivity natural frequencies base-rate neglect evidence
Bayesian decision rule expected loss type bluffing decision tree decision node chance node
backward induction expected value of perfect information (EVPI)
expected value of sample information (EVSI)

1.8 Test Yourself

1. Lottery M: 1,600 TL with probability 40%, 100 TL with probability 60%. Utility $u(x) = \sqrt{x}$. What are the expected value, the expected utility, the certainty equivalent, and the risk premium?

Between M and a sure 500 TL, which does this person choose?

Answer

$E[M] = 0.4 \cdot 1{,}600 + 0.6 \cdot 100 = 640 + 60 = 700$ TL. $E[u] = 0.4 \cdot 40 + 0.6 \cdot 10 = 16 + 6 = 22$. $CE = 22^2 = 484$ TL; risk premium $700 - 484 = 216$ TL. A sure 500 TL is greater than the CE ($\sqrt{500} \approx 22.4 > 22$); the person chooses the sure 500 TL. A risk-neutral person would choose M, because $700 > 500$.

2. A decision maker's utility function is u. Which of the following preserve all of her lottery preferences exactly: $v_1 = 4u + 7$, $v_2 = u^2$ (when $u \geq 0$), $v_3 = -2u + 1$, $v_4 = u/3 - 10$?

Answer

v_1 and v_4 preserve them: both are of the form $a \cdot u + b$ with a positive (4 and 1/3). v_3 is affine but $a = -2 < 0$; it reverses the whole ranking. v_2 is increasing but not affine; it preserves the order of sure outcomes but can change the order among lotteries (the square of $\sqrt{x}$ is x, and this reverses the decision between A and B).

3. Wealth is 2,500 (thousand TL); with probability 4% a loss of 2,400 occurs, leaving 100. Utility $u(x) = \sqrt{x}$. What is the highest premium the person is willing to pay for full insurance, and the lowest premium a risk-neutral insurer would ask?

Answer

Without insurance $E[u] = 0.96 \cdot 50 + 0.04 \cdot 10 = 48 + 0.4 = 48.4$. $CE = 48.4^2 = 2{,}342.56$. The highest premium is $2{,}500 - 2{,}342.56 = 157.44$. The expected loss is $0.04 \cdot 2{,}400 = 96$; the insurer asks at least this. Premiums between 96 and 157.44 benefit both sides. The 61.44 in between is the person's risk premium (expected wealth 2,404 − CE 2,342.56). Although the probability is small, because the loss is large the highest premium the person is willing to pay is about 1.64 times the expected loss.

4. The probabilities are unknown. Payoffs: A (Good 60, Fair 30, Bad −40), B (40, 20, 0), C (100, −10, −60). Which action do maximin, maximax, Laplace, Savage, and Hurwicz with $\alpha = 0.6$ choose?

Answer

Row minima −40, 0, −60 ⇒ maximin B. Maxima 60, 40, 100 ⇒ maximax C. Averages 16.7, 20, 10 ⇒ Laplace B. Column bests 100, 30, 0; regrets A (40, 0, 40), B (60, 10, 0), C (0, 40, 60); the largest are 40, 60, 60 ⇒ Savage A. Hurwicz: A $0.6 \cdot 60 + 0.4 \cdot (-40) = 20$, B $0.6 \cdot 40 + 0 = 24$, C $0.6 \cdot 100 + 0.4 \cdot (-60) = 36$ ⇒ C. The five criteria choose three different actions.

5. A disease has a prevalence of 2%, the test's sensitivity is 95%, and its false-positive rate is 5%. With natural frequencies in a population of 10,000, what is $P(sick \mid +)$? What is the probability that

someone whose test is negative is sick?

Answer

200 sick: 190 positive, 10 negative. 9,800 healthy: 490 positive, 9,310 negative. $P(\text{sick} \mid +) = 190/(190 + 490) = 190/680 \approx 27.9\%$. $P(\text{sick} \mid -) = 10/(10 + 9{,}310) = 10/9{,}320 \approx 0.11\%$. A negative result nearly rules out the disease; a positive result is still mostly a false alarm.

6. After the positive test in the previous question, the physician decides. The loss of unnecessary treatment of a healthy person is 40, and the loss of failing to treat a sick person is 160. What is the treatment threshold and what is the decision?

Answer

Treatment is chosen when $(1 - p) \cdot 40 < p \cdot 160$: $40 < 200p \Rightarrow p > 0.2$. Since $p \approx 0.279 > 0.2$, treatment is chosen. The expected losses: treat $(490/680) \cdot 40 \approx 28.8$, wait $(190/680) \cdot 160 \approx 44.7$.

7. A firm's new product brings +300 in a good market (probability 0.5) and −250 thousand TL in a bad market (0.5); if not launched, 0. (a) What are the best decision without information and the EVPI? (b) For a free survey, $P(\text{positive} \mid \text{good}) = 0.8$ and $P(\text{positive} \mid \text{bad}) = 0.3$. What is the EVSI?

Answer

(a) Launch: $0.5 \cdot 300 + 0.5 \cdot (-250) = 25 > 0 \Rightarrow$ launch, 25. With the oracle, $0.5 \cdot 300 + 0.5 \cdot 0 = 150$; $EVPI = 150 - 25 = 125$. (b) $P(\text{positive}) = 0.4 + 0.15 = 0.55$. $P(\text{good} \mid \text{positive}) = 0.4/0.55 = 8/11$; launching gives $8/11 \cdot 300 + 3/11 \cdot (-250) = 1650/11 = 150 \Rightarrow$ launch. $P(\text{good} \mid \text{negative}) = 0.1/0.45 = 2/9$; launching gives $2/9 \cdot 300 + 7/9 \cdot (-250) \approx -127.8 \Rightarrow$ skip (0). The value with the survey is $0.55 \cdot 150 + 0.45 \cdot 0 = 82.5$; $EVSI = 82.5 - 25 = 57.5 \le EVPI = 125$.

8. The opponent is a bluffer with probability 25%. In a hand, the bluffer raises with probability 60% and the honest player with probability 20%. What is the probability that the opponent is a bluffer after one raise, and (in another hand) after a second raise? If calling brings +20 against a bluffer and −40 against an honest player, after which observation is calling profitable?

Answer

First raise: $0.25 \cdot 0.6 / (0.15 + 0.75 \cdot 0.2) = 0.15/0.30 = 0.5$. Second raise: $0.5 \cdot 0.6 / (0.3 + 0.5 \cdot 0.2) = 0.3/0.4 = 0.75$. Calling is profitable when $20p - 40(1 - p) > 0 \Rightarrow p > 2/3$. After the first raise, since $0.5 < 2/3$, folding is better; after the second raise, $0.75 > 2/3$, so she calls ($E[\text{call}] = 15 - 10 = 5$).

1.9 Further reading

- S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach* (2020), Chapters 12–16: treats probability, Bayes' rule, utility theory, decision trees, and the value of information step by step from the point of view of artificial agents.
- H. Raiffa, *Decision Analysis* (1968): a classic introduction that presents decision trees and the value of sample information through business decisions.
- G. Gigerenzer and U. Hoffrage, "How to improve Bayesian reasoning without instruction: Frequency formats" (1995): shows with experiments why natural frequencies make Bayesian computation easier.
- D. Easley and J. Kleinberg, *Networks, Crowds, and Markets* (2010), Chapter 16: describes how Bayes' rule is used in information cascades, where people decide by observing one another.

1.10 References

1. M. Allais, "Le comportement de l'homme rationnel devant le risque: critique des postulats et axiomes de l'école américaine", *Econometrica* 21(4), 503–546, 1953.
2. D. Bernoulli, "Specimen theoriae novae de mensura sortis", *Commentarii Academiae Scientiarum Imperialis Petropolitanae* 5, 1738.
3. W. Casscells, A. Schoenberger, T. B. Graboys, "Interpretation by physicians of clinical laboratory results", *New England Journal of Medicine* 299, 999–1001, 1978.
4. D. Easley, J. Kleinberg, *Networks, Crowds, and Markets*, Cambridge University Press, 2010 (Chapter 16: information cascades and Bayes' rule).
5. D. Ellsberg, "Risk, ambiguity, and the Savage axioms", *Quarterly Journal of Economics* 75(4), 643–669, 1961.
6. G. Gigerenzer, U. Hoffrage, "How to improve Bayesian reasoning without instruction: Frequency formats", *Psychological Review* 102(4), 684–704, 1995.
7. H. Raiffa, *Decision Analysis: Introductory Lectures on Choices under Uncertainty*, Addison-Wesley, 1968.
8. S. Russell, P. Norvig, *Artificial Intelligence: A Modern Approach*, 4th ed., Pearson, 2020 (Chapters 12–16: uncertainty, Bayes, decision theory).
9. L. J. Savage, "The theory of statistical decision", *Journal of the American Statistical Association* 46, 55–67, 1951.
10. J. von Neumann, O. Morgenstern, *Theory of Games and Economic Behavior*, Princeton University Press, 1944.

PART I · CHAPTER 2

Games and Representations

To study a situation as a game, one must first describe it precisely: who is playing, what each player can choose, who knows what, and what each outcome is worth to whom. These notes begin with the basic concepts that answer these questions. They then present the two ways of writing down a game, the matrix (normal form) and the tree (extensive form), and classify games along six dimensions. Through the Prisoner's Dilemma and six classic 2×2 games they show the difference between equilibrium and efficiency. The guess-two-thirds game, combinatorial games, and sequential games show how the theory meets algorithms: minimax, the XOR rule for Nim, and backward induction. For minimax to be usable on large trees one needs alpha-beta pruning, move ordering, and depth-limited search, which are worked out in detail on tic-tac-toe and Connect Four.

LEARNING OUTCOMES

Concepts
Define player, strategy, payoff, information, rationality, and common knowledge on a worked example.

Representation and classification
Use the normal form (matrix) and the extensive form (tree); tell games apart by timing, agreements, total payoff, symmetry, information, and horizon.

Equilibrium and efficiency
Explain the Nash equilibrium intuitively; show that Betray is dominant in the Prisoner's Dilemma and that the equilibrium is not Pareto optimal.

Classic games
Find the pure equilibria of six classic 2×2 games using best responses; interpret their conflict and coordination structure.

Algorithms
Explain level-k reasoning; solve small games by hand and in code with minimax and the XOR strategy for Nim.

Sequential games
Find the subgame perfect equilibrium by backward induction; interpret non-credible threats and the difference between finite and infinite horizons.

Game-tree search
Apply minimax and alpha-beta pruning by hand and in code; explain the effect of move ordering, the evaluation function, and the horizon effect.

2.1 Core concepts

The parts that make up a game look simple, but every later computation rests on their definitions. If a concept is missing or wrongly defined, for example if strategy is confused with action, all the later computations come out wrong. For this reason the concepts are first made concrete on a small game and then written down formally.

2.1.1 A first game: guess two-thirds of the average

Everyone in a class secretly writes down a number between 0 and 90. The number may be an integer or a decimal. The average of all the guesses is computed, and two-thirds of that average is taken as the **target**. The player whose guess is closest to the target wins; if several players are equally close, the prize is shared. This is the **guess-two-thirds-of-the-average game**.

What makes the game interesting is that the right answer depends on what others will write. Someone thinking alone might say: "If the numbers are spread randomly between 0 and 90, the average will be 45, and two-thirds of 45 is 30; so I will write 30." But if others reason the same way, most people will write 30; then the average will be close to 30 and the target close to 20. Someone who anticipates this writes 20. Each step is a reply to what the others thought one step earlier.

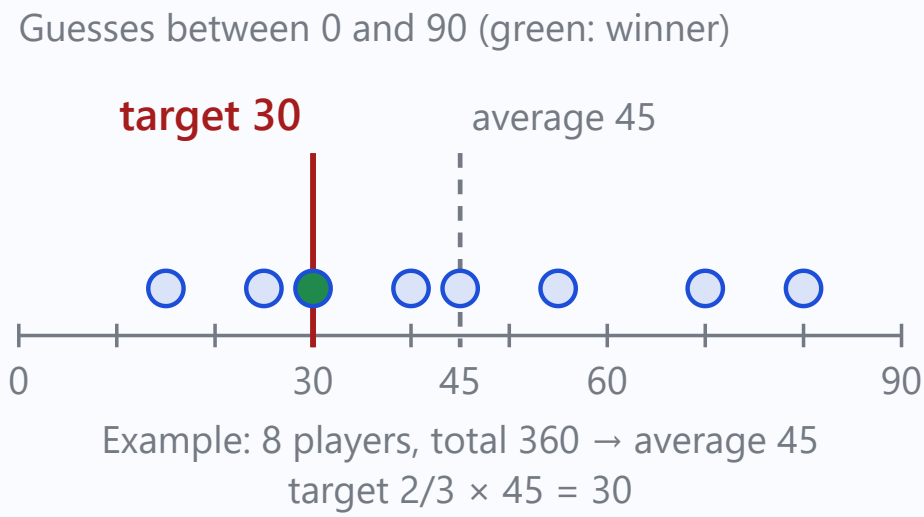


Figure 2.1: An example with eight players. The guesses sum to 360, so the average is 45; the target is 2/3 × 45 = 30. The guess closest to 30 (green) wins.

Figure 2.1 shows the guesses of eight players on a number line. The guesses sum to 360, so the average is 360 / 8 = 45. The target is two-thirds of 45, that is, 30. The player who wrote exactly 30 is the green dot and wins. Had the eight guesses been different, the average and the target would have been different too. The winning number comes not from the rules of the game but from the choices the players make together.

This small game contains almost all the questions of the chapter. Who are the players? What options does each player have? What does each choice earn whom? Do the players know how smart the others are? If "everyone is very smart", which number wins? In a real classroom, which number wins? The first four of these questions concern the *definition* of the game, the last two its *solution*. The solution is given in detail in Section 2.6.

2.1.2 What is game theory?

Game theory is a mathematical tool for studying situations in which decision makers are engaged in **strategic interaction**. Strategic interaction can be summed up briefly: one player's best decision depends on what the other does, and the other's best decision depends on the first's. In a single-person decision problem the decision maker plays only against nature or chance. In a game, on the other side are other decision makers who think for themselves and look after their own interests.

The decision makers are called **players** or **agents**. A player may be a person, a company, a country, a piece of software, or a network device. Game theory tries to explain how players behave and how the outcome takes shape. In doing so it assumes that players are **rational** and act according to their own interests. The exact meaning of this assumption is spelled out in Section 2.1.5.

The theory is widely used in economics, political science, biology, psychology, and computer science. Three example areas show what kinds of questions it answers:

- **Economics:** Price competition between firms, auctions, and bargaining. A firm's price takes on meaning only relative to its rival's price.
- **Computer networks:** Selfish routing, in which every packet or every driver chooses its own path. If everyone picks the fastest path, that path may become congested.
- **Artificial intelligence:** Multi-agent systems, game-playing programs, and learning agents. A chess program cannot choose a move without taking the opponent's best reply into account.

Algorithmic game theory adds two questions from computer science to classical game theory. The first concerns computation: how hard is it to compute the equilibrium of a game, and which algorithm does it? The second concerns design: how can one design rules, that is, algorithms, protocols, and auctions, that give good outcomes among selfish agents? These two questions are taken up again at the end of the chapter, in Section 2.10.

2.1.3 A short history

Game theory began early in the twentieth century with mathematical questions about games such as chess. By the middle of the century it had become one of the basic tools of economics. By the end of the century it had met computer science and turned into algorithmic game theory. Table 2.1 shows the main stops on this journey.

Year	Development
1913	Zermelo: in finite games of perfect information such as chess, either one side can force a win or both sides can force a draw.
1928	von Neumann: the minimax theorem for two-player zero-sum games.
1944	von Neumann and Morgenstern, *Theory of Games and Economic Behavior*: game theory becomes a field of its own.
1950	Nash: every finite game has an equilibrium in mixed strategies. In the same year the Flood–Dresher experiment is run at RAND, and Tucker tells the Prisoner's Dilemma story.
1961	Vickrey: in a second-price auction, bidding truthfully is dominant.
1994	Nobel Prize in Economics to Nash, Harsanyi, and Selten.
1999	Koutsoupias–Papadimitriou (the ratio of the worst equilibrium to the optimum, later named the Price of Anarchy) and Nisan–Ronen (algorithmic mechanism design): algorithmic game theory is born.
2006	No fast method is known for finding a Nash equilibrium in general games: the problem (exact equilibrium for two players, approximate equilibrium for many players) belongs to the class of hard problems called PPAD-complete (Daskalakis–Goldberg–Papadimitriou; two players: Chen–Deng).
2012, 2020	Nobel Prizes: Roth and Shapley (stable matching, market design); Milgrom and Wilson (auction design).

Table 2.1: Key developments from game theory to algorithmic game theory.

The dates in the table separate three periods. The years 1913–1944 are the period in which the mathematical foundations of the theory were laid. Zermelo's chess result and von Neumann's minimax theorem are the two great results of this period; both come back later: Zermelo in these notes (Section 2.2.8 and Section 2.7.1), the minimax theorem in Chapter 3 (Section 3.6.4). The years 1950–1994 are the period in which the theory spread to general games and to economics. Nash's equilibrium concept stands at its center. After 1999, two questions from computer science, "how hard is it to compute an equilibrium?" and "how are good rules designed?", moved to the center of the theory.

The birth of the Prisoner's Dilemma also appears in this table. In 1950 at RAND, Merrill Flood and Melvin Dresher had two people play the same game repeatedly, to test whether players behaved as the theory predicts. In the same year Albert Tucker described the structure of this game with a story about two prisoners, and the game became known by that name. The story is told in detail in Section 2.4.2.

2.1.4 The components of a game

A game is a situation in which players interact and each one's choice affects everyone's payoff. To write a situation down as a game, six questions must be answered. Table 2.2 lists these six components and their notation.

Component	Meaning	Notation or example
Players	The persons or institutions that make decisions.	$N = \{1, \ldots, n\}$
Actions / strategies	The set of actions each player can choose.	S_i = {Rock, Paper, Scissors}
Payoff	The value of each outcome to a player; it depends on everyone's choice.	$u_i(s_1, \ldots, s_n)$
Information	Who knows what, and when? Is the opponent's move observed?	chess ≠ poker
Rules and order	Simultaneous or sequential? The rules that link payoffs to actions.	normal / extensive form
Outcome	Results from all the players' choices taken together.	$s = (s_1, \ldots, s_n)$

Table 2.2: The components of a game and their notation.

The letter i in the notation means "any player". S_i is the set of strategies available to player i; u_i is the function that computes player i's payoff. The choices of all the players go into the payoff function, not only player i's. This is exactly what strategic interaction is: a player cannot determine their own payoff alone.

Example 2.1

The guess-two-thirds game in a class of 30 students breaks down into its components as follows:

1. **Players:** the 30 students in the class, $N = \{1, \ldots, 30\}$.
2. **Strategies:** for each student, all numbers between 0 and 90, $S_i = [0, 90]$. This set is infinite, because decimals are allowed.
3. **Outcome:** a list of 30 numbers, $s = (s_1, \ldots, s_{30})$.
4. **Payoff:** the average and the target are computed from the list. Those closest to the target share the prize equally; the others get 0. Student i's payoff depends on the whole list.
5. **Information:** everyone knows the rules, but nobody sees the numbers the others write.
6. **Rules and order:** everyone writes at the same time (secretly); the game is simultaneous.

This description also shows that the game is too large to draw as a matrix: each player has infinitely many strategies. Even so, the game is completely specified, and it is solved in Section 2.6.

2.1.5 Players, rationality, and common knowledge

Game theory makes three assumptions about the players. All three can be stated in plain language, but each has a subtle side.

Definition 2.1

- **Rationality:** Each player makes the choice that maximizes their own expected utility, given their beliefs about what the others will do.
- **Self-interest:** A player follows their own utility function. Other people's payoffs matter only if they enter the player's own utility.
- **Common knowledge:** A piece of information is common knowledge if everyone knows it, everyone knows that everyone knows it, everyone knows that this too is known to everyone, and this chain continues forever.

Two words in the definition of rationality matter. The first is "given their beliefs". A player does not know for certain what the others will do; they have a guess about it, that is, a belief. A rational player makes the best choice given this belief. If the belief is wrong the outcome may be bad, but the choice is still rational. The second is "expected utility". When the outcome is uncertain, the player weighs the utilities of the possible outcomes by their probabilities and adds them up. This computation is carried out in detail in Section 2.1.8.

The self-interest assumption is often misunderstood. **Rational does not mean selfish.** A utility function may include concerns for fairness, altruism, or reputation. Someone who voluntarily donates part of their earnings is rational if they derive utility from donating. The theory says only this: a player tries to do the best they can at whatever they care about. What is cared about is written into the utility function. The outcome, in turn, is the product of an interaction that no single player controls.

The common-knowledge assumption is the most abstract of the three; its effect is clearly visible in the guess-two-thirds game. Figure 2.2 shows which guesses each level of knowledge eliminates in this game.

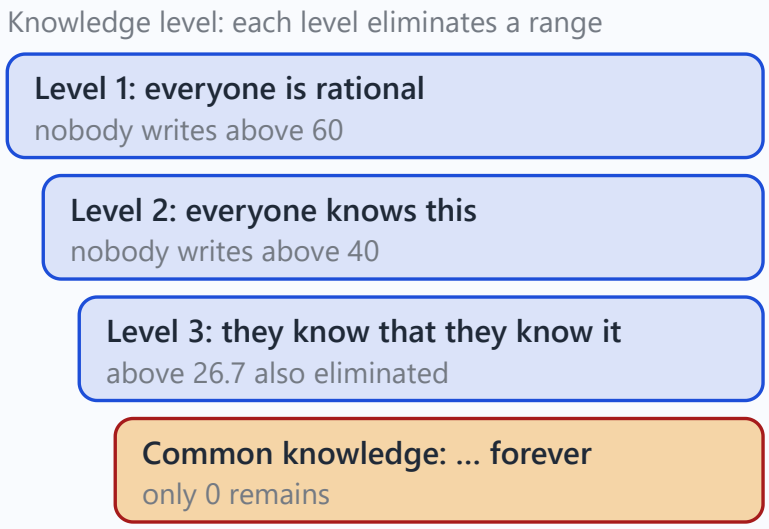


Figure 2.2: Levels of knowledge in the guess-two-thirds game. Each level eliminates a range: if everyone is rational, nobody writes more than 60; if everyone knows that, nobody writes more than 40; if they also know that they know it, everything above 26.7 is eliminated. Common knowledge (infinitely many levels) leaves only 0.

At level 1 there is only the knowledge that "everyone is rational". Since the average can be at most 90, the target is at most 60. Writing a number above 60 is never better than writing 60; a rational player does not write above 60. At level 2 everyone knows this: someone who knows that nobody will exceed 60 concludes that the average is at most 60 and the target at most 40. At level 3 the upper bound drops to 26.7. Each level adds one more "everyone knows" and multiplies the upper bound by 2/3. Common knowledge is this chain continuing forever, and in the end only 0 remains. This computation is carried out in code in Section 2.6.3.

Common mistake

- **"It is enough for everyone to know."** Everyone being rational is different from the rationality of everyone being common knowledge. At level 1 everyone is rational, yet only the range above 60 is eliminated. Reaching the result 0 requires infinitely many levels.
- **"A rational player never reduces their own money."** A utility function may include concerns other than money. Rationality means maximizing one's own utility; what the utility consists of varies from person to person.

2.1.6 Actions and strategies

In everyday language "action" and "strategy" are used interchangeably. In game theory the two are different, and the difference matters especially in sequential games.

Definition 2.2

An **action** is a single move a player can make at one decision point. A **strategy** is the plan of action a player chooses. In sequential games a strategy is a complete plan that says what the player will do *in every possible situation* they may face.

A list containing one strategy from each player is called a **strategy profile**: $s = (s_1, \ldots, s_n)$. The profile determines the outcome of the game.

In a simultaneous game each player decides once, without seeing the other. In this case strategy and action are the same thing: in Rock-Paper-Scissors "Rock" is both an action and a strategy. The difference appears when a player decides after seeing the other's move. Then the player's plan must contain a separate reply for every move they might see.

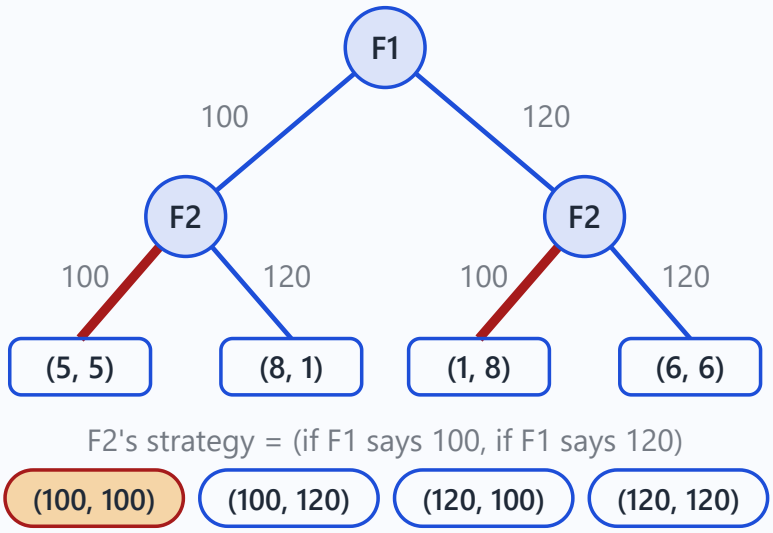


Figure 2.3: A sequential price game. Firm 1 (F1) chooses its price; Firm 2 (F2) sees that price and replies. Each strategy of F2 is written as a pair: (if F1 says 100, if F1 says 120). The bold branches show the (100, 100) plan.

In Figure 2.3, F2 sees F1's price and responds. F2 has two decision nodes: one where F1 said 100 TL, the other where F1 said 120 TL. At each node it chooses one of two actions (100 or 120). A strategy means making a choice at each of the two nodes. So F2 has $2 \cdot 2 = 4$ strategies: (100, 100), (100, 120), (120, 100), and (120, 120), although it has only 2 actions. For example, the strategy (120, 100) says: "If F1 says 100, I say 120; if F1 says 120, I say 100."

The general rule is this: if a player has k decision nodes and chooses among m actions at each, the number of strategies is m^k. If F2 had 3 nodes, the number of strategies would be $2 \cdot 2 \cdot 2 = 8$. This number grows exponentially with the number of nodes. For this reason writing large sequential games as matrices is impractical; the tree representation (Section 2.2.6) is far smaller.

Example 2.2

A student first decides whether to come to class. The teacher sees whether the student came and, in both cases, chooses between "assign homework" and "do not assign homework". If homework is assigned, the student says "do it" or "don't". The numbers of strategies of the teacher and the student are found as follows:

1. The teacher's decision nodes: "student came" and "student did not come". Two nodes with two actions each: $2^2 = 4$ strategies.
2. The student's decision nodes: the start (come / don't come), "I came and homework was assigned", and "I did not come and homework was assigned". Three nodes with two actions each: $2^3 = 8$ strategies.
3. One strategy of the student is, for example: (come; if homework is assigned after coming, do it; if homework is assigned after not coming, don't). The plan also covers the situation "I did not come and homework was assigned", which the student will never actually face.

The observation in the last item is important: a strategy makes a choice even for situations that are never reached. This detail becomes decisive when we examine non-credible threats in sequential games (Section 2.8.2).

A strategy rests on expectations about the others' behavior. When choosing a plan, a player predicts what the others will do. Besides this prediction, how the player makes the choice also matters. There are two kinds of strategy:

- **Pure strategy:** choosing a single plan with certainty. Example: "always Rock".
- **Mixed strategy:** a probability distribution over pure strategies. The player decides which pure strategy to play by rolling a die. Example: (Rock 1/3, Paper 1/3, Scissors 1/3).

A mixed strategy may look odd at first: why would a rational person leave the decision to a die? The answer is that in some games being predictable loses. In Rock-Paper-Scissors, a player who always says Rock loses every time once the opponent notices. Playing at random leaves no pattern for the opponent to exploit. This idea is shown with numbers in Section 2.2.3.

2.1.7 Payoffs and utility functions

A **payoff** is the value a player obtains, and it depends on the strategy profile chosen by all the players. The function that measures payoff is called the **utility function** and is written u_i for player i. Players choose their strategies to maximize their payoffs.

A payoff may be money, points, or a level of satisfaction. Sometimes a payoff is the negative of something bad: for costs such as years in prison or waiting time, less is better, so the payoff is written as −(years in prison). In this way the rule "larger is better" holds in all games. With mixed strategies the outcome is uncertain, so the player computes the **expected payoff**.

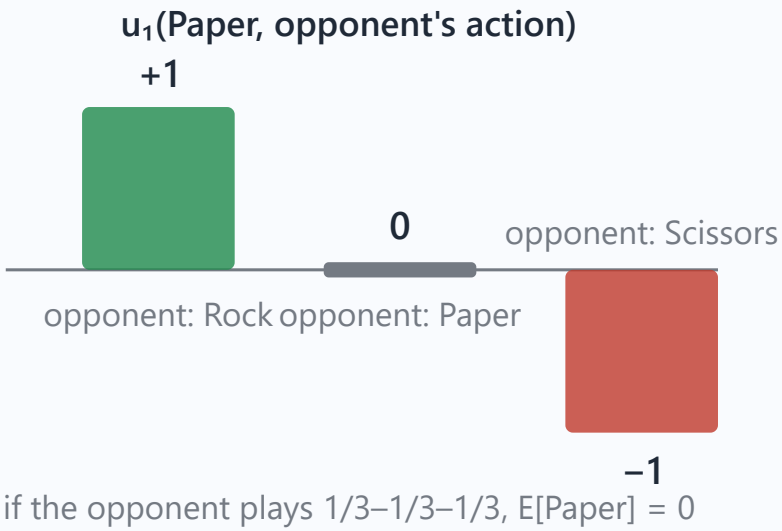


Figure 2.4: In Rock-Paper-Scissors, the payoff of a player who plays Paper depends on the opponent's action: +1 if the opponent plays Rock, 0 for Paper, −1 for Scissors. If the opponent plays each of the three actions with probability 1/3, the expected payoff of Paper is 0.

Figure 2.4 shows why a payoff is a function of a profile. Player 1 always chooses the same action, Paper. But the payoff is +1, 0, or −1 depending on the opponent's choice. The payoff therefore depends not on Paper alone but on the pair (Paper, opponent's action). In notation: $u_1(\text{Paper, Rock}) = +1$, $u_1(\text{Paper, Paper}) = 0$, $u_1(\text{Paper, Scissors}) = -1$.

Example 2.3

If the opponent plays each of the three actions with probability 1/3, the expected payoff of Paper is computed as follows. The payoff of each possible outcome is multiplied by its probability and the products are added:

$$E[u_1(\text{Paper})] = \tfrac{1}{3} \cdot (+1) + \tfrac{1}{3} \cdot 0 + \tfrac{1}{3} \cdot (-1) = 0$$

If the opponent played Rock more often, say (Rock ½, Paper ¼, Scissors ¼), the computation would change: $\tfrac{1}{2} \cdot 1 + \tfrac{1}{4} \cdot 0 + \tfrac{1}{4} \cdot (-1) = \tfrac{1}{4}$. The expected payoff of Paper would be positive, because Paper beats Rock. The expected payoff depends on both the player's own choice and the opponent's probabilities.

2.1.8 Ordinal and cardinal utility

Utility numbers can be used in two different senses. This difference determines which computations are meaningful.

Definition 2.3

Ordinal utility tells only the order of preference. If Ayşe's preferences are Opera ≻ Football ≻ Home (≻ means "is preferred to"), this order can be written as 3, 2, 0 or as 30, 20, 0; both carry the same information.

Cardinal utility, in contrast, also carries the size of the differences; it is also known as von Neumann–Morgenstern utility. Because uncertain outcomes are compared by expected utility, the numbers themselves matter.

With ordinal utility, changing the numbers without disturbing the order changes nothing. Even if 30, 20, 0 is written instead of 3, 2, 0, Ayşe still prefers Opera to Football and Football to staying home. Pure-strategy analysis, dominance, and pure equilibria look only at the question "which is better?", so they are not affected by such a change.

When uncertainty comes in, the situation changes. A **lottery** is an uncertain option written with the probabilities of its possible outcomes. "Opera with probability ½, Home with probability ½" is a lottery. The value of a lottery is the sum of the utilities of the outcomes weighted by probabilities, and this sum depends on the size of the numbers.

Example 2.4

Ayşe must decide between two options: Football for sure, or the lottery ½ Opera + ½ Home. For three different utility assignments, the decision is computed in Table 2.3.

Utility (Opera, Football, Home)	Lottery	Football for sure	Choice
(3, 2, 0)	½·3 + ½·0 = 1.5	2	Football
(3, 1, 0): same order	½·3 + ½·0 = 1.5	1	Lottery
2u + 1 = (7, 5, 1)	½·7 + ½·1 = 4	5	Football

Table 2.3: Three utility assignments with the same preference order, and the lottery decision.

1. In the first row the lottery gives 1.5 and Football for sure gives 2. Since 2 > 1.5, Ayşe chooses Football.
2. In the second row the preference order is the same (Opera ≻ Football ≻ Home); only the utility of Football has dropped from 2 to 1. The lottery is still 1.5, but Football for sure is now 1. Since 1.5 > 1, the decision switches to the lottery.
3. In the third row every utility of the first row has been multiplied by 2 and increased by 1: (3, 2, 0) → (7, 5, 1). The lottery gives 4 and Football for sure gives 5; the decision is Football, as in the first row.

Conclusion: even if the order stays the same, the lottery decision can change when the numbers change (rows 1 and 2). By contrast, a transformation of the form $a \cdot u + b$ ($a > 0$) preserves the decision (rows 1 and 3). This is because under such a transformation every expected value transforms in the same way: $\frac{1}{2}(2 \cdot 3 + 1) + \frac{1}{2}(2 \cdot 0 + 1) = 2 \cdot 1.5 + 1 = 4$. The difference between the two options is simply multiplied by a, and since $a > 0$ its sign does not change.

Common mistake

- **"A payoff of 4 is twice as good as a payoff of 2."** This reading is meaningful only for cardinal utility. If the numbers show only the order, 4 is better than 2, but how much better is unknown.
- **Using ordinal numbers in a mixed-strategy computation.** The expected-payoff computation assumes that the utilities are cardinal (vNM). If only the order is known, one settles for pure-strategy analysis.

2.1.9 Information: who knows what?

What the players know about the game, the payoffs, and the moves made directly affects strategy and outcome. In chess both players see the whole board. In poker the opponent's cards are hidden. This difference explains why playing well in the two games demands very different skills.

In a game tree, information is shown by an **information set**. An information set is a set of decision nodes that the player to move cannot tell apart. In the tree these nodes are joined by a dashed line. Because the player does not know which node they are at, they must choose the same action at every node in the set.

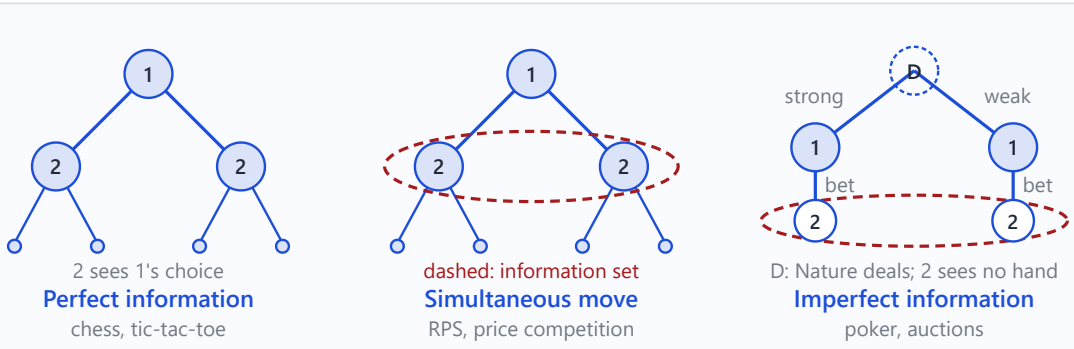


Figure 2.5: Three information situations. Left: player 2 sees player 1's choice (perfect information; chess, tic-tac-toe, Nim). Middle: two nodes lie in a dashed information set; player 2 cannot tell them apart (simultaneous moves; Rock-Paper-Scissors, price competition). Right: D (Nature) deals a card and player 2 does not see the hand (imperfect information; poker, auctions).

Figure 2.5 puts the three situations side by side. On the left every node is an information set by itself; player 2 sees player 1's choice and replies accordingly. In the middle player 2's two nodes are in the same information set. Player 1 has made a choice, but player 2 decides without seeing it; this is the same as the two players deciding at the same time. On the right, the first move is made by **Nature**: whether the hand is strong or weak is a chance event. Player 1 sees their own hand, player 2 does not. Player 2 sees that player 1 has bet but does not know whether the hand is strong or weak.

Definition 2.4

A game is a game of **perfect information** if every information set consists of a single node, that is, if each player sees all past moves when deciding. A game in which at least one information set contains more than one node is a game of **imperfect information**.

Perfect information means seeing the *past*, not knowing the future. A chess player does not know the opponent's next move, but sees every move played up to that point. This is why chess is a game of perfect information. Perfect information is also different from **complete information**, which means that everyone knows everyone's payoff functions. This distinction is taken up in Section 2.3.2.

2.1.10 Poker and bluffing

In Texas Hold'em poker each player has two hidden cards. In the middle are face-up cards that everyone can see, and each player tries to form the best hand by combining their hidden cards with the face-up ones. The opponent's hidden cards are not visible: the information is private. Even a player with a strong hand tries to guess the opponents' hands from their bets.

In this game the opponent's behavior, betting or waiting, carries information about their hand. A player who bets only with strong hands is easy to read: when they bet, the others fold; when they wait, the others attack. A **bluff** is sending a misleading signal to hide the strength of one's hand: betting with a weak hand too.

From the game-theoretic point of view, a bluff is a **mixed strategy**. Betting with a weak hand with a certain probability prevents the opponent from reading the player. If the probability is too low, the player is still read; if it is too high, the opponent starts calling every bet and bluffs become costly. The right probability is the point at which the opponent is indifferent between calling and folding. Computer poker programs play by computing such equilibria. A program called Libratus won a long series of matches against four professional players in 2017.

The poker example shows how information changes the structure of a game. In chess the notion of "the best move" is meaningful, because everything is open. In poker there is no single best move; the best behavior is a mixture that says what to do, and with what probability, in which hand.

2.2 Representing games

There are two basic ways to write down a game. The **normal form** writes the game as a table, that is, a matrix; it is natural when players decide at the same time. The **extensive form** draws the game as a tree; it shows the order of moves and who knows what at each point. The same game can be written in both ways.

2.2.1 Normal form

Definition 2.5

A **normal-form game** (also called a strategic-form game) $G = (N, (S_i), (u_i))$ consists of three components:

- $N = \{1, \ldots, n\}$: the set of players.
- S_i: the strategy set of player i.
- $u_i : S_1 \times \ldots \times S_n \to \mathbb{R}$: the utility function of player i. It assigns a real number to every strategy profile.

A strategy profile $s = (s_1, \ldots, s_n)$ contains one strategy from each player. The set of all profiles is written $S_1 \times \ldots \times S_n$.

The "×" sign in the definition is the **Cartesian product**: the Cartesian product of two sets is the set of all ordered pairs formed by one element of the first set and one element of the second. For example, if $S_1 = \{Top, Bottom\}$ and $S_2 = \{Left, Right\}$, then $S_1 \times S_2$ = {(Top, Left), (Top, Right), (Bottom, Left), (Bottom, Right)}. Each pair is a strategy profile. The definition of the utility function $u_i : S_1 \times \ldots \times S_n \to \mathbb{R}$ says: u_i takes a profile as input and returns a number. The input is the choice of all players, not only that of player i.

In a two-player game the normal form is written as a **payoff matrix**. The rows are the strategies of player 1 and the columns those of player 2. Each cell holds two numbers: first the row player's payoff, then the column player's, that is, (u_1, u_2). This order does not change anywhere in these notes.



Figure 2.6: A two-player normal-form game. S_1 = {Top, Bottom}, S_2 = {Left, Right}. For the highlighted profile s = (Top, Right), $u_1 = 1$ and $u_2 = 2$.

Figure 2.6 makes this reading concrete. The game is written $G = (N, S_1, S_2, u_1, u_2)$ with $N = \{1, 2\}$. There are four profiles, and each is a cell. The highlighted cell (Top, Right) = (1, 2) means: when player 1 chooses Top and player 2 chooses Right, player 1 earns one unit and player 2 earns two. Player 1 can choose only the row; player 2's column determines which cell the game lands in.

Example 2.5

In the game of Figure 2.6, the values of the two utility functions for the four profiles are:

1. $u_1(Top, Left) = 4$, $u_2(Top, Left) = 3$.
2. $u_1(Top, Right) = 1$, $u_2(Top, Right) = 2$.
3. $u_1(Bottom, Left) = 3$, $u_2(Bottom, Left) = 1$.
4. $u_1(Bottom, Right) = 2$, $u_2(Bottom, Right) = 4$.

If player 1 knew that player 2 would play Left, they would choose Top (4 > 3). If they knew player 2 would play Right, they would choose Bottom (2 > 1). So player 1's best choice depends on player 2; the table alone does not say "player 1 should play this". This dependence is the starting point of the best-response concept in Section 2.4.1.

The normal form models situations in which players decide **simultaneously**, or decide without knowing each other's choice. Two firms announcing their prices in sealed envelopes on the same day is an example. What matters is not time but information: even if one decides in the morning and the other in the evening, the game is simultaneous if the one deciding in the evening does not see the morning decision.

2.2.2 Rock-Paper-Scissors in normal form

In Rock-Paper-Scissors (RPS), two players ($n = 2$) simultaneously choose Rock, Paper, or Scissors. Rock beats Scissors, Scissors beats Paper, Paper beats Rock. The winner gets +1, the loser −1, and in a tie both get 0.

		Player 2		
Player 1	Rock	Paper	Scissors	
Rock	(0 , 0)	(−1 , 1)	(1 , −1)	Winner +1 Loser −1 Draw 0 Sum is 0 in every cell (zero-sum game)
Paper	(1 , −1)	(0 , 0)	(−1 , 1)	
Scissors	(−1 , 1)	(1 , −1)	(0 , 0)	

Figure 2.7: The payoff matrix of Rock-Paper-Scissors. The winner gets +1, the loser −1, and a tie gives 0; in every cell the two payoffs sum to 0.

In the matrix of Figure 2.7, the diagonal cells (Rock, Rock), (Paper, Paper), (Scissors, Scissors) are ties: (0, 0). In the other cells one player wins and the other loses. For example (Paper, Rock) = (1, −1): the row player beats Rock with Paper. In every cell the two numbers sum to 0.

Looking carefully at the matrix, one more thing becomes visible: in no cell are both players content with their choice. One of them always wants to change. In the cell (Rock, Scissors) = (1, −1) the column player wins by switching to Paper. In the cell (Paper, Paper) = (0, 0) each player can win by switching alone to Scissors. None of the nine cells passes this "nobody wants to change" test. This observation says that RPS has no equilibrium in pure strategies. The equilibrium concept is defined in Section 2.4.1.

2.2.3 Choosing a strategy in Rock-Paper-Scissors

What rule should a player who plays RPS over and over follow? Three possible rules can be compared.

1. Random choice. In each round every symbol is chosen with probability 1/3. This strategy cannot be exploited at all: whatever the opponent does, the expected payoff is 0. For example, if the opponent always says Rock, the player's payoff is 0 with probability 1/3 (Rock), +1 with probability 1/3 (Paper), and −1 with probability 1/3 (Scissors); the average is 0. The same computation can be repeated for every choice of the opponent.

2. Choice based on the past. "If the opponent said Rock last round, say Paper this round." This rule wins against an opponent who plays in patterns. But the rule is itself a pattern. If the opponent notices it, they can beat the player by saying Rock last round and Scissors this round.

3. An unbalanced mixed strategy. For example, Paper half of the time and Scissors half of the time.

Example 2.6

The player plays ½ Paper, ½ Scissors. The expected payoff against each of the opponent's three pure options is computed separately (from the opponent's point of view):

1. If the opponent says Rock: −1 against Paper, +1 against Scissors. Expected $½ \cdot (−1) + ½ \cdot (+1) = 0$.
2. If the opponent says Paper: 0 against Paper, −1 against Scissors. Expected $½ \cdot 0 + ½ \cdot (−1) = −½$.
3. If the opponent says Scissors: +1 against Paper, 0 against Scissors. Expected $½ \cdot (+1) + ½ \cdot 0 = +½$.

If the opponent always says Scissors, their expected payoff is +1/2 and the player's is −1/2. This mixture can be exploited. The player is unpredictable, since nobody knows in which round they will say Paper; but because they never choose Rock, Scissors is safe against them.

Comparing the three rules contains a lesson: even in a simple game the choice of strategy matters. Being unpredictable is not enough; one must be unpredictable *with the right probabilities*. In RPS the right probabilities are 1/3 each, and they guarantee 0 against any opponent.

2.2.4 Zero-sum games

Definition 2.6

A two-player game is **zero-sum** if the payoffs sum to zero in every strategy profile: $u_1(s) + u_2(s) = 0$ for every s. If the sum is a constant other than zero, the game is **constant-sum**; the same analysis applies to such games.

In a zero-sum game one player's gain is the other's loss. RPS, Matching Pennies, and chess (with +1 / 0 / −1 scoring) are zero-sum. A game in which one point is divided between two players is also constant-sum: if one gets 0.7, the other gets 0.3. By contrast, the Prisoner's Dilemma and price competition are not zero-sum; in these games the total payoff varies with the outcome.

In a zero-sum game a single matrix suffices, because the column player's payoff is the negative of the row player's. The row player wants to increase their payoff; the column player wants to decrease the row player's payoff. Two natural criteria then arise. **Maximin:** the row player looks at the worst case of each row and chooses the row whose worst case is best. **Minimax:** the column player looks at the best outcome the row player can get in each column and chooses the column that makes it smallest. Minimax search in a game tree (Section 2.7.2) is the algorithm that applies the same idea, from the leaves to the root, in a sequential game.

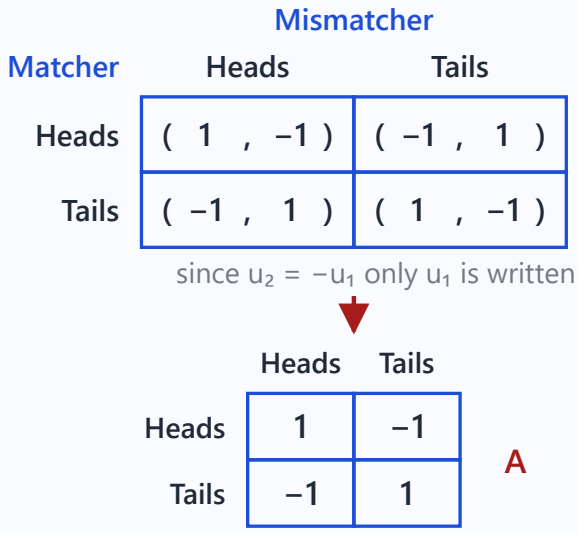


Figure 2.8: Matching Pennies. The Matcher wins 1 if the two coins show the same face, the Mismatcher if they show different faces. Since $u_2 = -u_1$, the matrix A, which lists only the row player's payoffs, describes the game completely.

Figure 2.8 shows this shorthand. The table at the top lists both payoffs; the matrix A below lists only the row player's payoff. The column player's payoff in each cell is the negative of the number in A. This shorthand cannot be used in games that are not zero-sum, since there one player's payoff cannot be derived from the other's.

Example 2.7

In Matching Pennies, $A = [[1, -1], [-1, 1]]$. Maximin and minimax in pure strategies are found as follows:

1. The row player's worst outcome in each row: in the Heads row $\min(1, -1) = -1$, in the Tails row $\min(-1, 1) = -1$. Maximin = $\max(-1, -1) = -1$.
2. The column player looks at the row player's best outcome in each column: in the Heads column $\max(1, -1) = 1$, in the Tails column $\max(-1, 1) = 1$. Minimax = $\min(1, 1) = 1$.
3. Maximin (-1) and minimax (1) are not equal. Two players using pure strategies can each guarantee themselves only the worst case.

The gap between the two values shows that the game has no pure "stable point". When we move to mixed strategies (½–½), both players guarantee an expected 0 and the gap closes. The minimax theorem that von Neumann proved in 1928 says that in mixed strategies these two values are equal in every finite zero-sum game.

In a zero-sum game there is no common interest; the game is pure competition. In games that are not zero-sum there can be outcomes in which everyone wins or everyone loses. Cooperation and competition coexist; communication and negotiation can change the outcome. Trade negotiations and climate agreements are games of this kind: if an agreement is reached both sides gain, and if not both lose.

2.2.5 The price competition game

Two firms sell a similar product in the same market and set their prices at the same time: 100 TL or 120 TL. The cheaper one captures the customers; if both charge the high price, the profit margin stays high. Table 2.4 gives the profits in units.

F1 \ F2	100 TL	120 TL
100 TL	**(5, 5)**	(8, 1)
120 TL	(1, 8)	(6, 6)

Table 2.4: The price competition game. Each cell shows the profits (Firm 1, Firm 2).

If both say 100 TL, each earns 5; if both say 120 TL, each earns 6 units of profit. If one says 100 and the other 120, the cheaper one gets most of the customers and earns 8; the expensive one earns only 1.

Example 2.8

From Firm 1's point of view, the two cases are examined separately:

1. If Firm 2 says 100 TL: Firm 1 earns 5 with 100 and 1 with 120. Since 5 > 1, 100 is better.
2. If Firm 2 says 120 TL: Firm 1 earns 8 with 100 and 6 with 120. Since 8 > 6, 100 is again better.
3. Whatever the other side does, 100 TL earns more. Such a strategy is called a **dominant strategy**.
4. Since the game is symmetric, the same computation holds for Firm 2: 100 TL is dominant for it too.

The outcome is (100, 100), and each firm earns 5. Yet (120, 120) would have given both 6. The individually best choice leads both to a worse outcome.

This structure is that of the Prisoner's Dilemma, which is examined in detail in Section 2.4. The price competition game is not zero-sum: the sums in the cells are 10, 9, 9, and 12. The cell with the largest sum is (120, 120), but the players cannot reach it.

2.2.6 Extensive form

The normal form does not show who plays first. In sequential interactions (chess, bargaining, market entry) the order matters: the second mover decides after seeing the first mover's move. The natural way to write such situations is the game tree.

Definition 2.7

The **extensive form** is a **game tree** that shows the order of decisions and who knows what at each point. The parts of the tree are:

- **Root:** the node where the game starts.
- **Decision node:** a point at which a player decides; the player to move is written on the node.
- **Branch:** an action.
- **Leaf** (terminal node): the point at which the game ends; each player's payoff is written there.
- **Information set:** nodes that the same player cannot tell apart (joined by a dashed line).

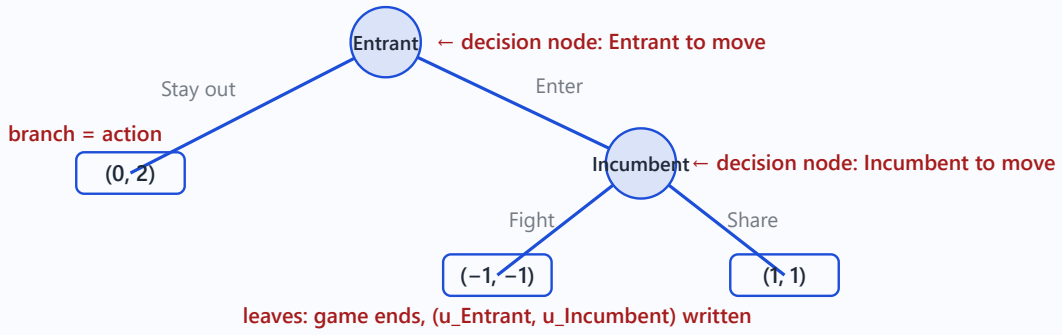


Figure 2.9: The tree of the market entry game. The Entrant decides whether to enter the market; if it enters, the Incumbent fights or shares. The leaves show $(u_{\text{Entrant}}, u_{\text{Incumbent}})$.

In Figure 2.9 the root is the Entrant's decision node. If the Entrant says "Stay out", the game ends: the Entrant earns 0 and the Incumbent keeps the market alone and earns 2. If the Entrant says "Enter", it is the Incumbent's turn. If the Incumbent says "Fight", a price war hurts both: (−1, −1). If the Incumbent says "Share", they split the market: (1, 1). The tree shows the order and who moves at each decision point. The solution of this game is given in Section 2.8.2.

2.2.7 From normal form to game tree

A simultaneous game can also be drawn as a tree. For this the tree uses an information set. The conversion, using the price competition game (Table 2.4), goes step by step as follows:

1. **Root:** Firm 1's decision. Each branch is an action: 100 TL or 120 TL.
2. **Second level:** at the end of each branch there is a decision node for Firm 2. It also chooses 100 or 120.
3. **Leaves:** there are four paths, and each corresponds to a cell of the matrix. For example the path 100–120 is the cell (8, 1).
4. **Information set:** because the two firms set their prices at the same time, F2 does not see F1's choice. F2's two nodes are placed in the same information set with a dashed line.

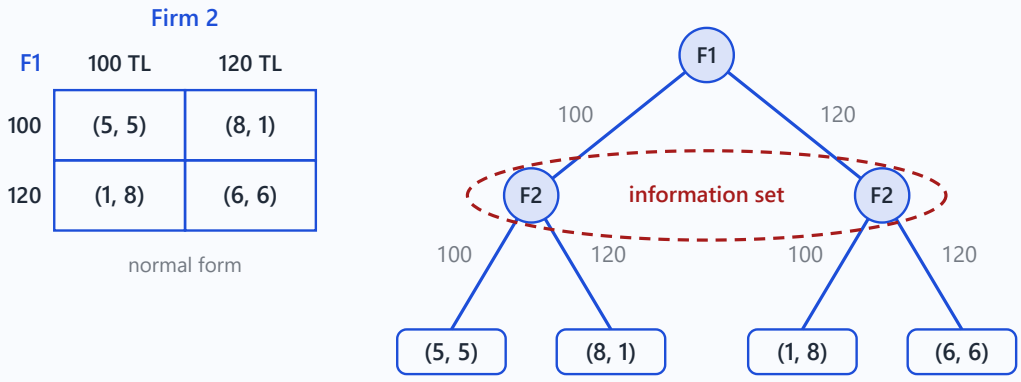


Figure 2.10: The matrix (left) and the tree (right) of the price competition game. F2's two nodes are in the same information set: F2 does not see F1's price. This tree is the same game as the matrix.

The tree in Figure 2.10 describes exactly the same game as the matrix. The tree draws F1 first, but this is not a time order; the information set says that F2 does not know F1's choice. Putting F1 at the bottom and F2 at the top instead would give the same game.

If the information set is removed, the game changes. Then F2 decides after seeing F1's price: the game is now sequential.

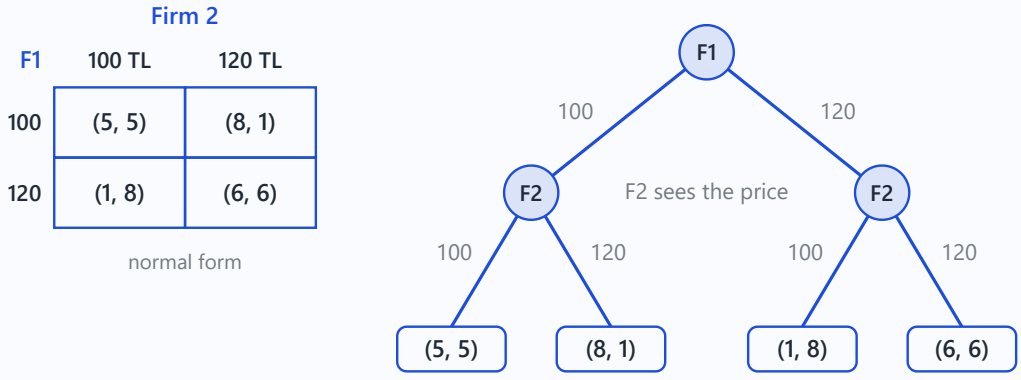


Figure 2.11: With the information set removed, F2 sees F1's price. The leaves are the same, but the game is now sequential and is solved by backward induction.

In the game of Figure 2.11 the leaves are the same, but the number of strategies of F2 has changed. In the simultaneous game F2 had two strategies (100 or 120), because it could not tell its two nodes apart and had to choose the same thing at both. In the sequential game the two nodes are separate and F2 has four strategies (Figure 2.3). Information directly changes the strategy set.

Example 2.9

The sequential price game is solved from the end backward in the tree of Figure 2.11:

1. If F1 said 100, F2's options are: 5 with 100, 1 with 120. F2 says 100; the node's value is (5, 5).
2. If F1 said 120, F2's options are: 8 with 100, 6 with 120. F2 says 100; the node's value is (1, 8).
3. F1, knowing this, compares: 5 with 100, 1 with 120. F1 says 100.

The outcome is again (100, 100) = (5, 5). F2's strategy is (100, 100), the plan "I say 100 whatever F1 says". The reason the order does not change the outcome in this game is that 100 is dominant for both players. In games without a dominant strategy, the order can change the outcome.

2.2.8 Chess is an extensive-form game

Chess is the best-known example of the extensive form. Two players move in turn; each move is a branch and each position is a node. The game has perfect information: both players see the board and all past moves. The result is a win, a loss, or a draw. A draw can arise from stalemate, from threefold repetition of a position, or from the fifty-move rule.

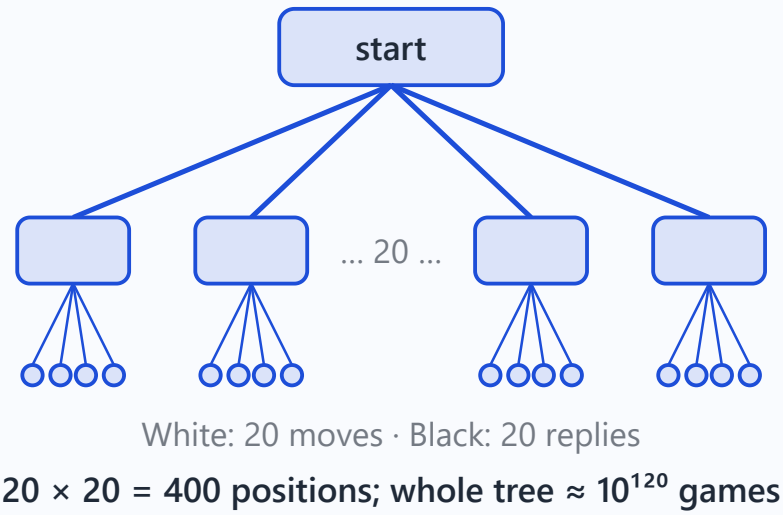


Figure 2.12: The first two levels of the chess tree. White has 20 opening moves and Black has 20 replies to

each: after two plies there are $20 \times 20 = 400$ positions. The whole tree contains about 10^{120} games.

In 1913 Ernst Zermelo proved a basic result for games of this kind.

Theorem 2.1 (Zermelo)

In a two-player, finite game of perfect information with no chance moves, either one player can guarantee a win, or both players can guarantee at least a draw.

The theorem says that the result of chess is "determinate": if both sides play without error, either White wins, or Black wins, or the game ends in a draw. But which one it is remains unknown, because the tree is so large. Figure 2.12 shows this size. After the first two plies there are 400 positions. With about 30 options per ply on average and games lasting dozens of moves, the number of games in the whole tree is about 10^{120}. This number is far larger than the number of atoms in the observable universe (about 10^{80}).

For this reason chess programs search only part of the tree. The search stops at a certain depth, the positions there are scored by an evaluation function, and unnecessary branches are pruned (minimax and pruning). The trees of small games such as tic-tac-toe, Nim, and "Don't Say 21" can be solved completely. These games are solved in Section 2.7.

2.3 Classifying games

Games are very diverse; instead of studying each one separately it is useful to group them by shared properties. The group a game falls into determines which solution method to use. For example, a sequential game of perfect information is solved by backward induction; a simultaneous game is solved on the matrix with best responses.

2.3.1 Six dimensions

Six questions are asked to classify a game. Each question is a dimension, and each dimension has two ends. Table 2.5 gives the six dimensions, the question asked in each, and an example for each end.

Dimension	One end	Other end	Question asked	Example
Timing	Simultaneous	Sequential	Do I play after seeing the opponent's move?	sealed-bid offer / market entry
Agreements	Cooperative	Non-cooperative	Can binding agreements be made?	joint venture / Prisoner's Dilemma
Total payoff	Zero-sum	Non-zero-sum	Is one player's gain the other's loss?	penalty kick / climate agreement

Dimension	One end	Other end	Question asked	Example
Symmetry	Symmetric	Asymmetric	Is the game the same if the players swap places?	Stag Hunt / Battle of the Sexes
Information	Perfect information	Imperfect information	Do I see all past moves?	Connect Four / poker
Horizon	Finite	Infinite	Does the game have a definite end?	Centipede / ongoing price competition

Table 2.5: Six dimensions for classifying games.

Timing. In a simultaneous game the players decide without seeing each other's choice; a sealed-bid auction is an example. In a sequential game one player moves after seeing the other's move; in the market entry game the Incumbent decides after seeing that the Entrant has entered. As seen in Section 2.2.7, the difference between the two kinds is an information set.

Agreements. In a cooperative game, players can make *binding* agreements: a promise can be enforced from outside, for example by a contract and a court. In a non-cooperative game there is no such enforcement; players may talk, but a promise is not binding. The unit of analysis also differs between the two kinds. In a cooperative game the question is: which group, that is, which **coalition**, acts together, and how does it share the payoff? The Shapley value and the core answer this question. In a non-cooperative game the unit of analysis is the single player and the central concept is the Nash equilibrium. The name "non-cooperative" does not mean that players never cooperate: cooperation can arise, but only if it is in each player's own interest.

Total payoff. In a zero-sum game (Section 2.2.4) one player's gain is the other's loss; a penalty kick is an example: a goal is a gain for the kicker and a loss for the goalkeeper. In a climate agreement there are outcomes in which everyone wins or everyone loses.

The **symmetry** and **horizon** dimensions call for a short formal definition. The **information** dimension was defined in Section 2.1.9: in Connect Four both players see the whole board; in poker the opponent's cards are hidden.

2.3.2 Complete information, symmetry, and horizon

Complete information is a notion separate from perfect information. Under complete information everyone's payoff function is known to everyone: each player knows what the others want and how much. If this is not known, the game has **incomplete information**. In an auction each bidder does not know the value the others place on the item; this is a case of incomplete information. Games of incomplete information are modeled as **Bayesian games**: each player's unknown attribute is thought of as drawn from a probability distribution.

The difference between the two notions can be summed up as follows. Perfect information is about *moves*: who played what in the past? Complete information is about *the game itself*: what are everyone's payoffs? Chess has both perfect and complete information. Rock-Paper-Scissors has

complete information (the payoffs are known to everyone) but not perfect information (the moves are made simultaneously).

Definition 2.8

A two-player game is **symmetric** if the two players' strategy sets are the same and, for every a, b,

$$u_1(a, b) = u_2(b, a)$$

holds. That is, the payoff depends not on who plays, but only on which strategy is played against which.

The equality in the definition must be read carefully: what the row player gets when the row player plays a and the column player plays b must equal what the column player gets when the roles are swapped (row plays b, column plays a). In the matrix this means that the first number in cell (a, b) equals the second number in cell (b, a).

Example 2.10

The price competition game (Table 2.4) is checked against the definition of symmetry. Four equalities are needed:

1. $u_1(100, 100) = 5$ and $u_2(100, 100) = 5$: equal.
2. $u_1(100, 120) = 8$ and $u_2(120, 100) = 8$: equal.
3. $u_1(120, 100) = 1$ and $u_2(100, 120) = 1$: equal.
4. $u_1(120, 120) = 6$ and $u_2(120, 120) = 6$: equal.

All four equalities hold, so the game is symmetric. In Battle of the Sexes, by contrast, (Opera, Opera) = (3, 2): $u_1(Opera, Opera) = 3 \neq u_2(Opera, Opera) = 2$. The definition fails at the first step; the game is asymmetric.

The **horizon** is how many rounds the game lasts. A game with a finite horizon has a definite end; the Centipede game (Section 2.8.5) ends in four moves. A game with an infinite horizon has no predetermined last round; two firms setting prices anew every month is such a game. With an infinite horizon, future payoffs are weighted by a **discount factor** $\delta \in (0, 1)$, and the player's total payoff is written as:

$$U = u_0 + \delta\, u_1 + \delta^2\, u_2 + \ldots = \Sigma_t\, \delta^t\, u_t$$

Here u_t is the payoff in round t. If δ is small, the future matters little: the player is impatient. If δ is close to 1, the future matters almost as much as the present. δ has a second interpretation as well: the probability that the game continues for one more round after each round. In this reading δ^t is the probability that the game reaches round t.

Example 2.11

A player earns 1 unit each round and $\delta = 0.9$. The total payoff is computed as follows:

1. The first few terms: $1 + 0.9 + 0.81 + 0.729 + \ldots$
2. This is a geometric series. Let the sum be S. If $S = 1 + \delta + \delta^2 + \ldots$, then $\delta S = \delta + \delta^2 + \ldots$. Subtracting the two equations gives $S - \delta S = 1$, that is, $S = 1 / (1 - \delta)$.
3. For $\delta = 0.9$, $S = 1 / 0.1 = 10$.

Even if infinitely many rounds are played, the total is finite, because the weight of distant rounds shrinks steadily. Ten units one period from now are worth $0.9 \cdot 10 = 9$ units today. This formula is used to compute when cooperation survives in the infinitely repeated Prisoner's Dilemma (Section 2.8.8).

2.3.3 Classifying a game

The six dimensions are independent of each other; a game is placed separately on each one. For chess the six answers are: sequential (players move in turn), non-cooperative (there are no binding agreements), zero-sum (if one wins the other loses), not symmetric (White moves first; the rules, however, are the same for both sides), perfect information (the board is open), and finite (the draw rules prevent the game from lasting forever).

Example 2.12

Three games from this chapter are classified along the six dimensions:

Dimension	Rock-Paper-Scissors	Prisoner's Dilemma	Market entry
Timing	Simultaneous	Simultaneous	Sequential
Agreements	Non-cooperative	Non-cooperative	Non-cooperative
Total payoff	Zero-sum	Non-zero-sum	Non-zero-sum
Symmetry	Symmetric	Symmetric	Asymmetric (different roles)
Information	Imperfect	Imperfect	Perfect information
Horizon	Finite (one round)	Finite (one round)	Finite

Table 2.6: Classification of three games.

Simultaneous games require an information set in the tree and therefore do not have perfect information. In the market entry game, by contrast, the Incumbent decides after seeing that the Entrant has entered; every information set is a single node. The sums of the outcomes in this game are 2, −2, and 2; since the sum is not constant, the game is not zero-sum. Because the Entrant and the Incumbent have different options and payoffs, the game is asymmetric.

Common mistake

- **Confusing perfect information with complete information.** Perfect information concerns past moves, complete information concerns payoff functions. RPS has complete information but not perfect information.
- **"If they can talk, the game is cooperative."** Talking is not enough; the agreement must be *binding*. Even if the prisoners promise each other before the interrogation, the game remains a non-cooperative game.
- **Checking symmetry by looking only at the diagonal.** It is not enough for the two numbers in the (a, a) cells to be equal; the off-diagonal cells (a, b) and (b, a) must also match crosswise.

2.4 Equilibrium and the Prisoner's Dilemma

Writing down a game is not the same as solving it. The "solution" of a game is a prediction of which strategies the players will choose. The most basic solution concept of game theory is the Nash equilibrium. The concept stays at an intuitive level here; the Prisoner's Dilemma then shows that equilibrium and efficiency are different things.

2.4.1 Best response and Nash equilibrium

Definition 2.9

The strategies of the players other than player i are written s_{-i}. With s_{-i} fixed, the strategy that gives player i the highest payoff is called a **best response**.

A profile in which everyone's strategy is a best response to the others' strategies is called a **Nash equilibrium** and is written s^*. At equilibrium nobody wants to deviate unilaterally (Nash, 1950).

The definition can be read as a "regret test". After the game is played, each player asks themselves: "If I had known what the others did, would I change my choice?" If everyone's answer is "no", the profile is a Nash equilibrium. If a single player's answer is "yes", it is not.

In a two-player matrix, best responses and equilibria can be visualized with arrows. From each cell an arrow is drawn in the direction in which one of the players wants to deviate. A cell from which no arrow leaves is an equilibrium.

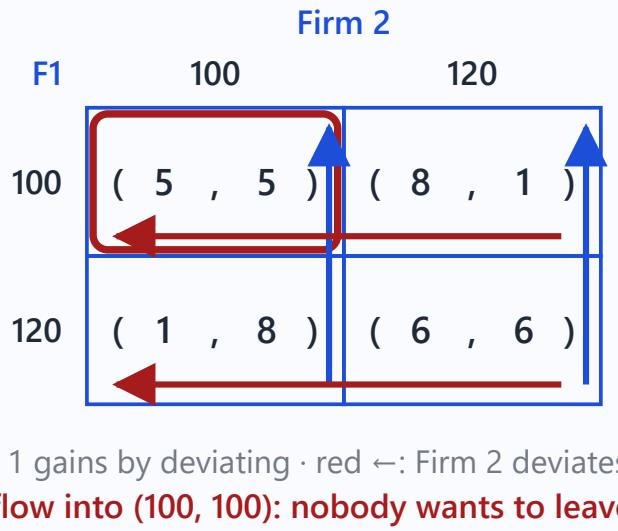


Figure 2.13: Deviation arrows in the price competition game. Blue arrows show Firm 1's (the row player's) profitable deviations, red arrows Firm 2's (the column player's). All arrows flow into (100, 100); no arrow leaves this cell.

Example 2.13

The arrows in Figure 2.13 are found for the four cells as follows:

1. (120, 120) = (6, 6): if F1 switches to 100 it gets 8 (8 > 6), an upward arrow. If F2 switches to 100 it gets 8, a leftward arrow. Two arrows leave.
2. (100, 120) = (8, 1): if F2 switches to 100 it gets 5 (5 > 1), a leftward arrow. If F1 switches to 120 it gets 6 (6 < 8); F1 does not deviate.
3. (120, 100) = (1, 8): if F1 switches to 100 it gets 5 (5 > 1), an upward arrow. F2 does not deviate (6 < 8).
4. (100, 100) = (5, 5): if F1 switches to 120 it gets 1 (1 < 5); if F2 switches to 120 it gets 1. No arrow leaves.

The only equilibrium is (100, 100). Deviation arrows flow into the equilibrium and none flows out of it.

What the Nash equilibrium *says* is this: with the others fixed, deviating does not pay; nobody regrets. An equilibrium is a self-sustaining situation. What the Nash equilibrium does *not* say is that the equilibrium is good for everyone. In price competition (120, 120) would have given both firms 6; the

equilibrium gives 5 each. The Prisoner's Dilemma has the same structure. Here the intuition behind the concept and its use in the classic examples are enough.

2.4.2 The Prisoner's Dilemma: story and matrix

Two suspects, Sam and John, are interrogated in separate rooms and cannot talk to each other. Each has two options: stay silent, or betray the partner by confessing. Table 2.7 gives the sentences in the four cases.

Case	Sam	John
Both stay silent	1 year	1 year
Sam confesses, John stays silent	free	3 years
Sam stays silent, John confesses	3 years	free
Both confess	2 years	2 years

Table 2.7: Prison terms in the Prisoner's Dilemma.

For consistency of terminology, two names are fixed. **Cooperation** means cooperating with the accomplice, that is, *staying silent*. **Betrayal** means confessing (cooperating with the police). In short, the two strategies are written "Silent" and "Betray".

Years in prison are bad; fewer is better. The payoff is written as −(years in prison): a light sentence means a large payoff. Table 2.8 gives the payoff matrix of the game. In each cell the first number is Sam's (row) payoff and the second is John's (column) payoff, and the cells are named a, b, c, d.

Sam \ John	Silent	Betray
Silent	a (−1, −1)	b (−3, 0)
Betray	c (0, −3)	**d (−2, −2)**

Table 2.8: The payoff matrix of Sam and John. Payoff = −(years in prison).

The table is read as follows. If both stay silent (cell a), both get −1. If Sam stays silent and John betrays (cell b), Sam gets −3 and John 0: Sam serves 3 years and John goes free. Cell c is the reverse. If both betray (cell d), both get −2. The basic question is this: what happens if each prisoner wants to minimize their own sentence?

2.4.3 Why is Betray dominant?

Sam does not know what John will do. So he examines the two cases separately. This method is called **case analysis**: a best response is found for each possible choice of the opponent.

Example 2.14

1. **Case 1: John stays silent.** If Sam stays silent he gets −1 (cell a); if he betrays he gets 0 (cell c). Since 0 > −1, betraying is better.
2. **Case 2: John betrays.** If Sam stays silent he gets −3 (cell b); if he betrays he gets −2 (cell d). Since −2 > −3, betraying is again better.
3. Whatever John does, betraying brings Sam a lighter sentence. Betray is a **dominant strategy** for Sam.
4. The game is symmetric; John faces the same dilemma and Betray is dominant for him too.
5. Both betray. The outcome is cell d: (−2, −2), that is, two years in prison each.

Definition 2.10

A strategy s_i of player i is a **(strictly) dominant strategy** if it gives strictly more than any other strategy against every possible choice of the others: $u_i(s_i, s_{-i}) > u_i(s_i', s_{-i})$ for every $s_i' \neq s_i$ and every s_{-i}.

If a dominant strategy exists, the player does not have to predict what the other will do. Sam does not need to know John's probabilities; the same answer is best in every case. This is the strongest feature of a dominant strategy: it requires no belief about the opponent.

Common mistake

- **"Betray is chosen because it reduces the average sentence."** This reasoning is wrong. Betray is better against *every* choice of John; there is no need to average or to know John's probabilities.
- **"Confessing is the best outcome for both."** It is not. If both had stayed silent (cell a) they would have got (−1, −1), which is better for both than (−2, −2). The individually best choice makes both worse off.
- **Confusing sentences with payoffs.** If years in prison are read directly as payoffs, "choosing the larger" would mean choosing 3 years. Never compute without putting the minus sign.

2.4.4 The message of the dilemma

The game's only Nash equilibrium is (Betray, Betray) = (−2, −2). This is easy to check: from cell d, if Sam switches to Silent he gets −3, and if John switches to Silent he gets −3; neither deviates. In the other three cells at least one player gains by switching to betrayal. By contrast, (Silent, Silent) = (−1,

−1) is better for both. So the equilibrium is worse than an outcome that makes both better off; in the language to be defined in Section 2.4.5, the equilibrium is **not Pareto optimal**.

The situation does not change even if the prisoners agree before the interrogation. The game is a non-cooperative game: the agreement is not binding. After the promise "we will both stay silent", Sam, believing John will stay silent, betrays and gets 0 (instead of −1). For both of them, breaking the promise pays. Since the promise does not change the payoffs, it does not change the dominant strategy either.

The message of the game is this: everyone behaving rationally in their own interest does not guarantee a good outcome for society. This shows that the idea of the "invisible hand", that behavior pursuing individual interest spontaneously leads to social good, does not always hold.

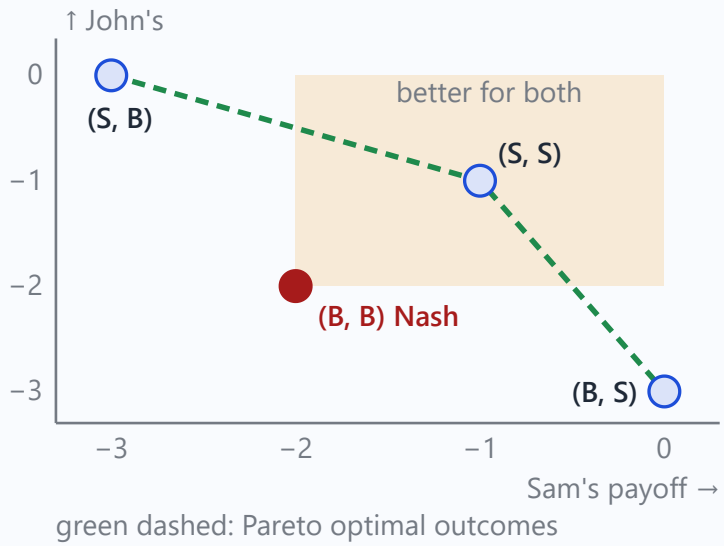


Figure 2.14: The four outcomes of the Prisoner's Dilemma in payoff space. The horizontal axis is Sam's payoff, the vertical axis John's. The Nash equilibrium (B, B) is red; (S, S) in the orange region is better for both. The green dashed line joins the Pareto optimal outcomes.

The structure of the Prisoner's Dilemma appears often in everyday life:

- **Price wars:** every firm cuts its price, and in the end all profits fall (Table 2.4).
- **Arms races:** every country arms itself, nobody is safer, and all become poorer.
- **Climate agreements:** every country wants to avoid the cost of cutting emissions.
- **Doping in sport:** if everyone dopes, the relative ranking does not change, but everyone's health suffers.
- **Overuse of common resources:** fishermen catch more even though they know the stock will run out.

There are also ways out of the dilemma. The first is **repeated play and reputation**: if the same players meet again and again, today's betrayal can be punished tomorrow. The tit-for-tat strategy cooperates in the first round and then copies the opponent's previous move. How cooperation survives under infinite repetition is computed in Section 2.8.8. The second is **contract and incentive design**: if the payoffs are changed, for example by adding a penalty for betrayal, betrayal stops being dominant. Designing the rules of the game is the subject of mechanism design (Section 2.10.3).

2.4.5 Pareto optimality

The Nash equilibrium is a concept of *stability*. To measure whether an outcome is *efficient*, a separate concept is needed. The weakest and most common criterion for this is Pareto optimality.

Definition 2.11

An outcome x **Pareto dominates** an outcome y if and only if $u(x) \geq u(y)$ for every player and $u(x) > u(y)$ for at least one player.

An outcome that no outcome Pareto dominates is **Pareto optimal**: nobody can be made better off without making someone worse off.

In payoff space, the plane in which each outcome is drawn as a point (u_1, u_2), the definition has an easy visual counterpart. The upper-right quadrant of a point is the region of points that give both players at least as much. If this region contains no other outcome (apart from the point itself), the point is Pareto optimal.

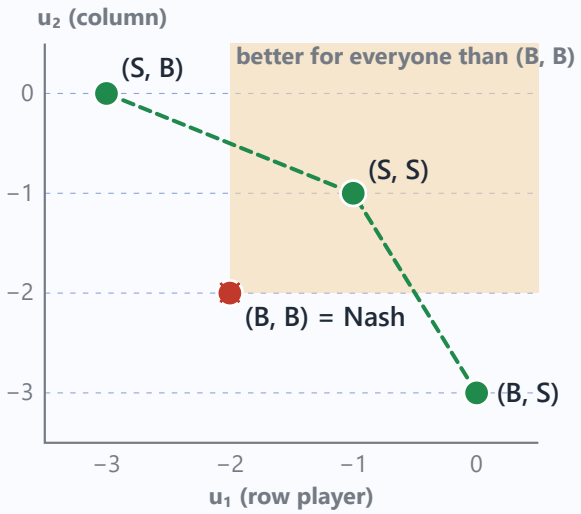


Figure 2.15: Pareto optimality in payoff space. In the upper-right quadrant (orange) of the point (B, B) = (−2, −2) lies (S, S); the Nash equilibrium is Pareto dominated. The three green points (S, B), (S, S), (B, S) are Pareto optimal.

Example 2.15

The four outcomes of the Prisoner's Dilemma (S: Silent, B: Betray) are checked one by one:

1. (B, B) = (−2, −2): (S, S) = (−1, −1) gives both more (−1 > −2). It is Pareto dominated.
2. (S, S) = (−1, −1): to dominate it we would need an outcome that gives both players at least −1 and one of them more. (S, B) = (−3, 0) makes Sam worse off, (B, S) = (0, −3) makes John worse off. It is Pareto optimal.
3. (S, B) = (−3, 0): no outcome gives John more than 0; this is the only outcome that gives 0. Nobody can dominate it. It is Pareto optimal.
4. (B, S) = (0, −3): by the same reasoning it is the only outcome that gives Sam 0. It is Pareto optimal.

Conclusion: (S, S), (S, B), and (B, S) are Pareto optimal; (B, B), the only Nash equilibrium, is Pareto dominated. The game's equilibrium is the only inefficient outcome among the four.

Pareto optimality does not mean fairness. (S, B) is Pareto optimal, but Sam serves 3 years while John goes free. Pareto optimality says only "nothing is wasted"; it does not look at how the payoff is distributed. An outcome in which one player takes everything and the other nothing can also be Pareto optimal.

Equilibrium (stability) and efficiency are separate questions. The Prisoner's Dilemma shows that the two questions can have different answers: what is stable is not efficient, and what is efficient is not stable. The loss in between is measured by the **Price of Anarchy**. If years in prison are counted as costs, the total cost at equilibrium is 2 + 2 = 4 years and at the best outcome (S, S) it is 1 + 1 = 2 years; the ratio is 4 / 2 = 2. Selfish behavior doubles the total cost.

2.5 Classic 2×2 games

Games in which two players each have two strategies are called **2×2 games**. Their matrices consist of only four cells, but these small games represent very different strategic situations: pure conflict, a problem of trust, a problem of meeting, and a contest of nerve. Six classic games cover all of these situations.

2.5.1 Six games and their pure equilibria

Table 2.9 writes the six games in a single format. Each row is one game. The column (1, 2) shows the cell where the row player plays their 1st action and the column player their 2nd action; in each cell the row player's payoff is written first, then the column player's.

Game	Actions (1 / 2)	(1, 1)	(1, 2)	(2, 1)	(2, 2)	Pure Nash equilibria
Matching Pennies	Heads / Tails	(1, −1)	(−1, 1)	(−1, 1)	(1, −1)	none
Chicken	Swerve / Straight	(0, 0)	(−1, 1)	(1, −1)	(−10, −10)	(Swerve, Straight), (Straight, Swerve)
Battle of the Sexes	Opera / Football	(3, 2)	(0, 0)	(0, 0)	(2, 3)	(Opera, Opera), (Football, Football)
Stag Hunt	Stag / Hare	(4, 4)	(0, 3)	(3, 0)	(3, 3)	(Stag, Stag), (Hare, Hare)
Pure Coordination	Party / Home	(10, 10)	(0, 0)	(0, 0)	(5, 5)	(Party, Party), (Home, Home)
Free Money	Heads / Tails	(1, 1)	(0, 0)	(0, 0)	(0, 0)	(Heads, Heads) strict; (Tails, Tails) weak

Table 2.9: Six classic 2×2 games and their pure Nash equilibria.

There is a systematic way to find pure equilibria: **best-response marking**. In each column the row player's largest payoff is marked; this is the row player's best response to that column. In each row the column player's largest payoff is marked; this is the column player's best response to that row. A cell carrying both marks is a cell in which each player's choice is a best response to the other's, that is, a pure Nash equilibrium. If there are ties, all of the tied largest values are marked.

Example 2.16

In Chicken, two drivers approach each other on a narrow road. Each either swerves to the side (Swerve) or goes straight (Straight). The marking goes as follows:

1. Column "Swerve" (the column player swerves): the row player gets 0 with Swerve and 1 with Straight. Straight is best (1 > 0); a row mark goes on the cell (Straight, Swerve).
2. Column "Straight": the row player gets −1 with Swerve and −10 with Straight. Swerve is best (−1 > −10); a row mark goes on the cell (Swerve, Straight).
3. Row "Swerve": the column player gets 0 with Swerve and 1 with Straight. Straight is best; a column mark goes on the cell (Swerve, Straight).
4. Row "Straight": the column player gets −1 with Swerve and −10 with Straight. Swerve is best; a column mark goes on the cell (Straight, Swerve).

1 \ 2	Swerve	Straight
Swerve	(0, 0)	**(−1*, 1*)**
Straight	**(1*, −1*)**	(−10, −10)

Table 2.10: Marking in Chicken. The green cells carry both marks, that is, they are pure Nash equilibria.

The cells with two marks are (Swerve, Straight) and (Straight, Swerve). The game has two pure equilibria, and in both one driver swerves and the other goes straight. (Straight, Straight) is the worst outcome: a collision.

Example 2.17

In Matching Pennies, marking leaves no cell with two marks:

1. Column Heads: the row player gets 1 with Heads and −1 with Tails → a row mark on (Heads, Heads).
2. Column Tails: the row player gets −1 with Heads and 1 with Tails → a row mark on (Tails, Tails).
3. Row Heads: the column player gets −1 with Heads and 1 with Tails → a column mark on (Heads, Tails).
4. Row Tails: the column player gets 1 with Heads and −1 with Tails → a column mark on (Tails, Heads).

The row marks are on the diagonal and the column marks are off the diagonal; no cell carries two marks. There is no pure Nash equilibrium. In every cell the losing player wants to deviate, and the deviation arrows trace a cycle.

Example 2.18

In Free Money, if both players say Heads each gets one unit; in the other three cells nobody gets anything. Marking requires care with ties:

1. Column Heads: the row player gets 1 with Heads and 0 with Tails → Heads is marked.
2. Column Tails: the row player gets 0 with Heads and 0 with Tails → a tie; *both* are marked.
3. Row Heads: the column player gets 1 with Heads and 0 with Tails → Heads is marked.
4. Row Tails: the column player gets 0 with Heads and 0 with Tails → both are marked.

The cells with two marks are (Heads, Heads) and (Tails, Tails). The two equilibria are of different kinds. (Heads, Heads) is a **strict equilibrium**: a player who deviates drops from 1 to 0, that is, loses. (Tails, Tails) is a **weak equilibrium**: a player who deviates goes from 0 to 0; deviating does not gain, but it does not lose either. Heads **weakly dominates** Tails: it is never worse and is better in at least one case.

The marking for the remaining three games is done the same way. In Battle of the Sexes, (Opera, Opera) = (3, 2) and (Football, Football) = (2, 3) carry two marks; in the off-diagonal (0, 0) cells both players want to move to the meeting point. In Stag Hunt, (Stag, Stag) = (4, 4) and (Hare, Hare) = (3, 3)

are equilibria: someone who knows the other is hunting Stag gets 4 from Stag and 3 from Hare; someone who knows the other is hunting Hare gets 0 from Stag and 3 from Hare. In Pure Coordination, (Party, Party) and (Home, Home) are equilibria.

Common mistake

- **Thinking one mark is enough.** For a cell to be an equilibrium, *both* players must be playing a best response. In a cell carrying only a row mark, the column player deviates.
- **Searching in the wrong direction.** The row player's best response is searched *within a column* (up and down), because the row player can change only their row. The column player's is searched *within a row* (left and right).
- **Looking at the wrong number.** For the row player the first number of each cell is compared, for the column player the second.
- **Picking one in a tie.** All of the tied largest values are marked; otherwise the weak equilibrium in Free Money is missed.

2.5.2 Conflict or coordination?

The six games line up on a spectrum according to how much the two players' interests overlap. At one end is Matching Pennies, where interests are completely opposed; at the other end are Pure Coordination and Free Money, where there is no conflict at all.

Matching Pennies: pure conflict. Interests are completely opposed and the game is zero-sum. In every cell the loser deviates. Any predictable tendency of a player is exploited: if the Matcher says Heads a little more often, the Mismatcher shifts to Tails. The only way out is to play at random, that is, the mixed strategy (½–½). The penalty kick is a real-life example of this structure: the kicker and the goalkeeper try to guess each other.

Chicken: anti-coordination. The players want to choose *different* actions: one must swerve and the other go straight. There are two equilibria, and each favors one driver. Who wins? The one who credibly shows that they cannot back down. For example, a driver who throws the steering wheel out of the window shows that they have no option other than Straight; the other driver's best response is then to swerve. Such behavior is called **commitment**. Commitment gains by reducing options; in a single-person decision problem reducing options never helps, but in a game it can.

Battle of the Sexes: coordination and conflict together. Two people want to spend the evening together; meeting is a shared interest. But one prefers Opera (3 > 2) and the other Football (3 > 2); where to meet is the conflict. There is no dominant strategy. Solutions are communication ("I'm going to the opera tonight"), convention, or taking turns (opera one week, football the next).

Stag Hunt: a trust dilemma. If two hunters hunt a stag together they get a big reward (4, 4). If one hunts a hare alone, they get 3; the other, chasing the stag, is left empty-handed (0). (Stag, Stag) is both an equilibrium and Pareto optimal. But Hare guarantees 3 in every case: whatever the other does, the

hare hunter gets 3. Stag requires trusting the other. The difference from the Prisoner's Dilemma is this: in Stag Hunt cooperation is an equilibrium. Nobody wants to deviate unilaterally from cooperation; the problem is not the wish to deviate but the fear that the other will deviate (Skyrms, 2004).

Pure Coordination: a bad equilibrium is lasting too. There is no conflict of interest: both players prefer to meet at the Party (10 > 5). But (Home, Home) is an equilibrium too: someone who knows the other will stay home gets 0 by going to the party and 5 by staying home. The bad equilibrium sustains itself. Players often head for what is salient, "what will occur to everyone". Thomas Schelling called such choices **focal points** (Schelling, 1960). For example, two people in a city who have not been told where to meet will often go to the best-known square of the city.

Free Money: a weak equilibrium. (Heads, Heads) is a strict equilibrium: a deviator loses. (Tails, Tails) is a weak equilibrium: deviating does not gain, but it does not lose either. Heads is weakly dominant; a sensible player has no reason to pick Tails. This game shows that the "≥" sign in the definition of the Nash equilibrium sometimes admits implausible equilibria too.

Example 2.19

Which of the six games are zero-sum and which are symmetric is checked against the definitions:

1. **Zero-sum:** the total must be 0 in every cell. In Matching Pennies all totals are 0. In Chicken the total of (−10, −10) is −20; the others also have cells whose total differs from zero (e.g. (4, 4) in Stag Hunt). Only Matching Pennies is zero-sum.
2. **Symmetric:** $u_1(a, b) = u_2(b, a)$ must hold. In Chicken $u_1(Swerve, Straight) = -1 = u_2(Straight, Swerve)$, $u_1(Straight, Swerve) = 1 = u_2(Swerve, Straight)$, and the diagonals are equal: it is symmetric. Stag Hunt, Pure Coordination, and Free Money pass the same check.
3. **Not symmetric:** in Battle of the Sexes $u_1(Opera, Opera) = 3 \neq 2 = u_2(Opera, Opera)$. In Matching Pennies $u_1(Heads, Heads) = 1 \neq -1 = u_2(Heads, Heads)$; one role wants to match and the other to mismatch.

Conclusion: only Matching Pennies is zero-sum; the symmetric ones are Chicken, Stag Hunt, Pure Coordination, and Free Money.

Game	Structure	Number of pure equilibria	Core problem
Matching Pennies	Pure conflict	0	Being unpredictable
Chicken	Anti-coordination	2	Who will back down? (commitment)
Battle of the Sexes	Coordination + conflict	2	Where to meet?

Game	Structure	Number of pure equilibria	Core problem
Stag Hunt	Trust	2	Will the other be trusted?
Pure Coordination	Purely shared interest	2	Avoiding the bad equilibrium (focal point)
Free Money	Shared interest, weak equilibrium	2	Choosing the weakly dominant strategy

Table 2.11: The structure of the six games at a glance.

2.6 The guess-two-thirds game

The game at the start of the chapter (Section 2.1.1) can now be solved. The game has two "answers". One is the theory's answer: what happens if everyone is fully rational and this is common knowledge? The other is the practical answer: what is the best guess in a real classroom? The two answers differ.

2.6.1 Level-k thinking

Real players do not think infinitely many steps; they stop after a few. One way to model this behavior is **level-k reasoning**. The model classifies players by how many steps they think.

Definition 2.12

A **level-0** player does not think strategically and writes a random number; on the range 0–90 the average of these guesses is 45. A **level-k** player ($k \geq 1$) assumes that everyone else is level $k - 1$ and best-responds to this assumption: they write two-thirds of the level-$k - 1$ guess.

Level of thinking	Assumption	Guess
Level 0	Everyone writes at random, average 45	— (random)
Level 1	Others are level 0	$2/3 \cdot 45 = 30$
Level 2	Others are level 1	$2/3 \cdot 30 = 20$
Level 3	Others are level 2	$2/3 \cdot 20 \approx 13.3$
Level ∞	Everyone is fully rational	0

Table 2.12: Guesses under level-k thinking.

The reasoning "the average will be 45, two-thirds of that is 30" is only level 1. Someone who thinks the others will also say 30 writes a lower number, for example 20. Someone who thinks everyone will write too low may shift back up a little. The level-k guess is in general $45 \cdot (2/3)^k$, and it approaches 0 as k grows.

The important point is this: *the best guess depends on the belief about what level the others think at*. If most of the class is level 1, the best response is level 2; thinking further loses. The notion of level should not be confused with the knowledge level (Figure 2.2), which counts the "everyone knows" layers of common knowledge. The level is how many steps a player thinks; the knowledge level is what is known about the players.

2.6.2 Code trace: the level-k guess

The Python code below, given a belief about the distribution of levels in the class, computes how far from the target each level will fall.

```
def guess(k, L0=45):     # level-k
    g = L0
    for _ in range(k):
        g = g * 2 / 3
    return g
share = {0: 0.3, 1: 0.4, 2: 0.3}   # belief
avg = sum(p * guess(k) for k, p in share.items())
target = avg * 2 / 3
for k in range(5):
    print(k, round(guess(k), 1),
          round(abs(guess(k) - target), 1))
```

The code works as follows:

- `guess(k, L0=45)` returns the guess of a level-`k` player. `g` starts at 45, and the loop runs `k` times, multiplying `g` by 2/3 each time. For `k = 0` the loop never runs and 45 is returned: the average of level 0.
- The dictionary `share` holds the belief: 30% of the class is level 0, 40% is level 1, 30% is level 2.
- `avg` is the expected average: each level's guess is multiplied by that level's share and the products are summed.
- `target` is two-thirds of the average; this is the winning number.
- The last loop prints, for each level from 0 to 4, the guess and its distance from the target.

Example 2.20

1. Level guesses: `guess(0) = 45`, `guess(1) = 30`, `guess(2) = 20`.
2. Expected average: $0.3 \cdot 45 + 0.4 \cdot 30 + 0.3 \cdot 20 = 13.5 + 12 + 6 = 31.5$.
3. Target: $2/3 \cdot 31.5 = 21$.
4. The distance of each level is listed in Table 2.13.

k	guess(k)	|guess − target|	Comment
0	45	24	writing the average is far too high
1	30	9	the classic "45 → 30" answer, still far
2	**20**	**1**	**closest to the target among the levels examined (the best response is the target itself, ≈ 21)**
3	13.3	7.7	thinking one level too far loses
4	8.9	12.1	the distance grows as the guess nears 0

Table 2.13: The output of the code (target = 21).

Against this population, the player who writes 20 wins. The player who writes 0, the Nash equilibrium answer, is 21 units away and loses. The best response depends on beliefs.

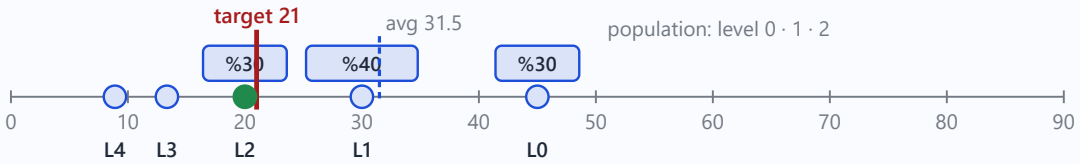


Figure 2.16: Level-k guesses (L0–L4) and the target. The population is 30% level 0, 40% level 1, and 30% level 2; the expected average is 31.5 and the target is 21. The guess closest to the target is that of level 2 (20).

Figure 2.16 shows the same result on a number line. The target (21) is very close to the level-2 guess (20). Levels 3 and 4 fall to the left of the target, levels 0 and 1 to the right. The location of the target depends on the population: if a larger part of the population were level 0, the target would shift to the right.

2.6.3 Iterated elimination

The theory's answer is found by a different method. This method deletes, step by step, the guesses that can never be good.

Definition 2.13

A strategy is **dominated** if another strategy gives a better result against every choice of the others. A rational player does not play a dominated strategy. **Iterated elimination of dominated strategies** means deleting the dominated strategies, then searching for newly dominated ones in the shrunken game, and continuing until no new deletion can be made.

In the 2/3 game the elimination works as follows. Initially every guess is between 0 and 90 and nothing has been eliminated.

1. **Round 1.** The average can be at most 90, so the target is at most 60. Every guess above 60 is never better than writing 60: if the target is below 60, then 60 is closer to it. These guesses are eliminated. (This elimination rests on *weak* domination: since the prize goes to the winner, 61 sometimes does as well as 60; for example with two players, if the opponent says 90 both win.)
2. **Round 2.** Everyone is rational and everyone knows it: nobody exceeds 60. Average ≤ 60 ⇒ target ≤ 40. Everything above 40 is eliminated.
3. **Round 3.** Average ≤ 40 ⇒ target ≤ 26.7. To take this step one must know that "everyone knows that everyone is rational".
4. **Rounds 4–6.** Average ≤ 26.7 ⇒ target ≤ 17.8; average ≤ 17.8 ⇒ target ≤ 11.9; average ≤ 11.9 ⇒ target ≤ 7.9.

In each round the upper bound is multiplied by 2/3; after k rounds the upper bound is $90 \cdot (2/3)^k$. The code below repeats this multiplication for six rounds.

```
hi = 90                # largest possible guess
for t in range(1, 7):
    # avg ≤ hi  ⇒  target ≤ (2/3)·hi
    hi = hi * 2 / 3
    print(t, round(hi, 1))
# hi = 90·(2/3)^k → 0 : unique equilibrium, all 0
```

t	Previous hi	New hi = hi · 2/3	Meaning
1	90	60	everyone is rational
2	60	40	+ everyone knows this
3	40	26.7	+ they know that they know it
4	26.7	17.8	one more layer
5	17.8	11.9	one more layer
6	11.9	7.9	one more layer

Table 2.14: Trace of the elimination code.

In the code the variable `hi` holds the "largest meaningful guess". In each round of the loop the inference in the comment line is applied: if the average cannot exceed `hi`, the target cannot exceed `(2/3)·hi`. The new `hi` is this value. The loop stops after 6 rounds, but in theory the process goes on forever.

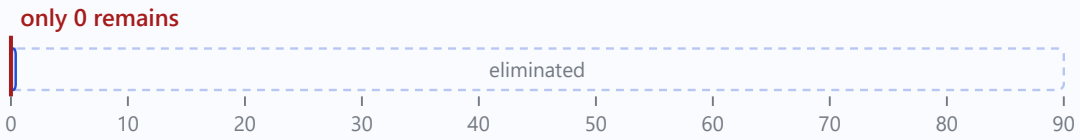


Figure 2.17: The end of iterated elimination. In each round the upper bound is multiplied by 2/3 and converges to 0; the only guess that survives elimination is 0.

The upper bound shrinks geometrically: each round it is two-thirds of the previous one. After how many rounds does the bound fall below 1? The inequality $90 \cdot (2/3)^k < 1$ first holds for $k = 12$ ($90 \cdot (2/3)^{12} \approx 0.69$; for $k = 11$ it is ≈ 1.04). But in no finite number of rounds is exactly 0 reached. The process continues forever and the upper bound converges to 0: the only profile that survives iterated elimination is everyone saying 0 (Figure 2.17).

2.6.4 The unique Nash equilibrium: everyone picks 0

Iterated elimination leaves the profile in which everyone says 0. That this profile is a Nash equilibrium is checked separately.

Example 2.21

Let all n players write 0. The regret test is applied:

1. If everyone says 0, the average is 0 and the target is 0. Everyone hits the target exactly and the prize is shared.
2. Let one player deviate to $x > 0$. The total becomes x, the average x/n, and the target $2x/(3n)$.
3. The others' 0 is at distance $2x/(3n)$ from the target. The deviator's x is at distance $x - 2x/(3n) = x(3n - 2)/(3n)$ from the target.
4. For $n \geq 2$, $3n - 2 > 2$, so the deviator's distance is larger. For example, for $n = 10$, $x = 30$ the target is 2; 0 is 2 away from the target and 30 is 28 away.
5. The deviating player loses and is also deprived of their share. The deviation is unprofitable; the profile is a Nash equilibrium.

That the equilibrium is unique cannot be derived from iterated elimination: the elimination rests on weak domination, and deleting weakly dominated strategies can also delete some equilibria. Uniqueness is shown directly. Let S be the sum of the others' guesses. If a player writes $g = 2S/(3n - 2)$, the new target is exactly g; the player hits the target exactly.

- In an equilibrium everyone must get a share of the prize: a player with a zero share would hit the target by changing their guess to g and get a share.

- If everyone gets a share, everyone is at the same distance from the target. If this distance is positive, a player who deviates to g wins alone and takes the whole prize that was being shared; this is a profitable deviation.
- What remains is the profile in which everyone is on the target: everyone says the same x and $x = (2/3)x$, that is, $x = 0$.

Hence everyone saying 0 is the only equilibrium. Elimination gives the intuitive way of finding this equilibrium, and the direct proof gives its uniqueness.

One detail shows how the rules of the game affect the equilibrium. If only integers were allowed, "everyone says 1" would also be an equilibrium with $n \geq 4$ players. Allowing decimal guesses removes this second equilibrium.

Example 2.22

Only integers are allowed, and all n players write 1. The average is 1, the target is 2/3, and everyone shares (each gets $1/n$ of the prize).

1. Let one player deviate to 0. The average becomes $(n - 1)/n$ and the target $2(n - 1)/(3n)$.
2. The deviator's distance is $2(n - 1)/(3n)$; the others' distance is $1 - 2(n - 1)/(3n) = (n + 2)/(3n)$.
3. The deviator wins alone only if $2(n - 1) < n + 2$, that is, $n < 4$. At $n = 4$ the two distances are equal (6/12 = 6/12) and the prize is again divided among four; the deviation does not raise the payoff. For $n > 4$ the deviator loses.
4. Let one player deviate to 2. The average becomes $(n + 1)/n$ and the target $2(n + 1)/(3n)$; for $n \geq 2$ this number is at most 1. 1 is closer to the target than 2 is; the deviator loses.

So for $n \geq 4$, "everyone says 1" allows no profitable deviation and is an equilibrium. With decimal guesses, however, in the situation "everyone says 1" a player deviating to a number such as 0.7 gets closer to the target (with 10 players the target becomes about 0.65; 0.7 is 0.05 away from it and 1 is 0.35 away) and wins alone; the equilibrium is destroyed.

2.6.5 Theory and experiment

The theory's answer is 0. But in experiments the winning number is not 0. In the 0–100 version run with *Financial Times* readers in 1997, the winning guess was 13. In Rosemarie Nagel's laboratory experiments, published in 1995, first-round guesses clustered around 1–2 steps of thinking: most guesses were near the guesses of level 1 or level 2 (Nagel, 1995). As the game was repeated with the same group, guesses came down toward 0.

Theory: iterated elimination	Largest meaningful guess	Reality: level-k	Guess
start	90	level 0 (random, average 45)	45
step 1	2/3 × 90 = 60	level 1 (others are level 0)	30

Theory: iterated elimination	Largest meaningful guess	Reality: level-k	Guess
step 2	$2/3 \times 60 = 40$	level 2 (others are level 1)	20
step 3	26.7	level 3 (others are level 2)	13.3
step 4	17.8	…	…
…	$\to 0$	level ∞	0

Table 2.15: The theory's answer and the steps of thinking in reality.

The two columns answer different questions. The left column shows what each layer of common knowledge eliminates: each step adds one more layer of the assumption "everyone is rational", and the end of infinitely many steps is the unique Nash equilibrium. The right column models what players actually write. The difference between theory and experiment comes from the fact that the assumption "everyone is fully rational and this is common knowledge" does not hold fully in reality. If even a few people write high numbers, the average rises and the player who writes 0 loses.

In this game, determining the Nash equilibrium is not hard: the equilibrium is unique and in pure strategies. What is hard is predicting the best response against real players. To win, one needs to know not 0 but how many steps the class thinks. If the class plays level 1 and the average comes out at 30, the target is 20; level 2 wins.

Example 2.23

Let the game be on the range 0–100 with the target "half of the average". Iterated elimination proceeds as follows:

1. Average at most 100 $\Rightarrow$ target at most 50. Everything above 50 is eliminated.
2. Average at most 50 $\Rightarrow$ target at most 25. Everything above 25 is eliminated.
3. The next bounds are 12.5; 6.25; 3.125; … In each round the bound is halved.

The upper bound converges to 0 and the unique equilibrium is again everyone saying 0. As long as the multiplier is below 1 (2/3, 1/2, …) the result does not change; only the speed of convergence does. The smaller the multiplier, the faster the bound shrinks.

2.7 Combinatorial games

Games such as chess, tic-tac-toe, and Nim share a structure: two players move in turn, everything is open, and there are no dice or cards. These games are the most "algorithmic" corner of game theory: in principle each of them can be solved by sweeping the tree from the end backward. This sweep takes three forms: minimax, backward labeling, and Nim's XOR rule.

2.7.1 Combinatorial games and Zermelo's theorem

Definition 2.14

A **combinatorial game** has the following properties: two players move in turn; information is perfect; there is no chance; the game ends in a finite number of steps.

Normal play: the player who makes the last move wins; the player who cannot move loses.

Nim is played under normal play: whoever takes the last token wins. In Don't Say 21 the rule is reversed: the player who says 21, that is, who is forced to make the last move, loses. In tic-tac-toe and chess a draw is also possible. Zermelo's theorem (Theorem 2.1) applies to all of these games: in such a game either one player can force a win or both can force a draw.

In some games, who wins can be found without searching the tree at all. Hex is a famous example. In Hex there are no draws: when the board is full, one of the players has necessarily connected their own sides. The **strategy-stealing argument** works as follows. Suppose the second player has a winning strategy. The first player places a random stone and then, putting themselves in the second player's place, plays that winning strategy. The extra stone on the board never hurts them; if the square the strategy asks for is already occupied, they place another random stone. Thus the first player wins too. Since it is impossible for both players to win at the same time, the assumption is false: the second player has no winning strategy. Since there are no draws, the first player wins. The argument does not say *how* the first player wins; it only shows that they do.

In combinatorial games every position is of one of two types:

Definition 2.15

In a game played under normal play, a position is an **N-position** (*next*: the player to move wins) if the player to move can guarantee a win, and a **P-position** (*previous*: the previous player wins) if they cannot. The labels are determined by two rules:

- *Every* move from a P-position goes to an N-position.
- *At least one* move from an N-position goes to a P-position.

The final position, in which no move can be made, is a P-position under normal play: the player to move cannot move and loses.

The logic of the rules is this. If the player to move can send the opponent to a P-position, they do so; whatever the opponent does, it leaves them an N-position again, and each time the player sends the opponent back to P. In the end it is the opponent who cannot move. If all of a player's moves go to N-positions, then whatever they do they leave the opponent a winning position.

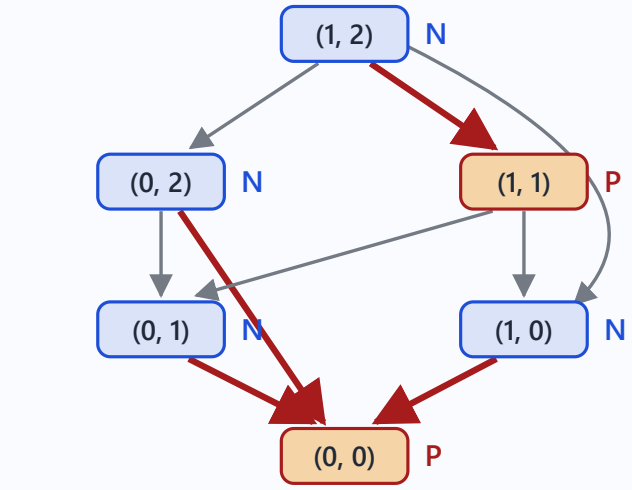


Figure 2.18: Two-heap Nim (1 and 2 tokens): all positions and their labels. P (red) is a position in which the player to move loses, N one in which the player to move wins. Bold arrows are winning moves.

Example 2.24

In two-heap Nim the heaps contain 1 and 2 tokens. The player to move takes at least one token from a single heap; whoever takes the last token wins. Positions are written (first heap, second heap) and the labels are computed from the leaves to the root (Figure 2.18):

1. (0, 0): no move. The player to move loses → **P**.
2. (0, 1) and (1, 0): the single move goes to (0, 0), which is P → **N**.
3. (0, 2): one can go to (0, 1) or to (0, 0). (0, 0) is P → **N**. The winning move is to take both tokens at once.
4. (1, 1): the moves go to (0, 1) and (1, 0); both are N → **P**.
5. (1, 2): the moves are (0, 2) [N], (1, 1) [P], (1, 0) [N]. Since (1, 1) is P → **N**. The winning move is to take one token from the second heap, leaving (1, 1).

The starting position (1, 2) is N: the first player wins if they play correctly. Computing the labels from the leaves to the root is the special case of minimax that works with win/lose values.

2.7.2 Code trace: minimax on a game tree

If the payoff values are numbers rather than just "win / lose", the same idea is carried out by the **minimax** algorithm. The two players are called MAX and MIN. MAX wants a large value and MIN wants a small one. The numbers at the leaves are MAX's payoff; since the game is zero-sum, MIN's

payoff is the negative of that. The value of a node is defined as follows: at a leaf it is the payoff at the end of the game; at a MAX node it is the largest of the children's values, and at a MIN node the smallest.

```
T = {"A": ["B", "C", "D"], "B": [3, 12, 8],
     "C": [2, 4, 6], "D": [14, 5, 2]}
def minimax(n, maxi):
  if isinstance(n, int):         # leaf
    return n
  best = float("-inf") if maxi else float("inf")
  for c in T[n]:
    v = minimax(c, not maxi)
    best = max(best, v) if maxi else min(best, v)
  return best
print(minimax("A", True))
```

The parts of the code are:

- The dictionary `T` holds the tree. The children of the root A are B, C, D. The children of B, C, D are leaves, represented by integers (`int`).
- `minimax(n, maxi)` returns the value of node `n` . If `maxi` is true, it is MAX's turn.
- If `isinstance(n, int)` is true, `n` is a leaf and its value is itself; the function returns immediately.
- `best` starts at −∞ for MAX and +∞ for MIN. These values mean "no child seen yet": the first value MAX sees is larger than −∞, and the first value MIN sees is smaller than +∞. (Fixed numbers such as −99 and 99 give wrong results when leaf values exceed these bounds.)
- The loop evaluates each child in turn. At a child it is the other player's turn; so the recursive call is made with `not maxi` . MAX updates `best` with `max` , MIN with `min` .

Step	Node (to move)	Value seen	best
1	B (MIN)	start	+∞
2	B	leaf 3	min(+∞, 3) = 3
3	B	leaf 12	min(3, 12) = 3
4	B	leaf 8	min(3, 8) = 3 → returns B = 3
5	A (MAX)	B = 3	max(−∞, 3) = 3
6	C (MIN)	leaves 2, 4, 6	+∞ → 2 → 2 → 2; C = 2
7	A (MAX)	C = 2	max(3, 2) = 3
8	D (MIN)	leaves 14, 5, 2	+∞ → 14 → 5 → 2; D = 2
9	A (MAX)	D = 2	max(3, 2) = 3 → result 3

Table 2.16: Trace of `minimax("A", True)` .

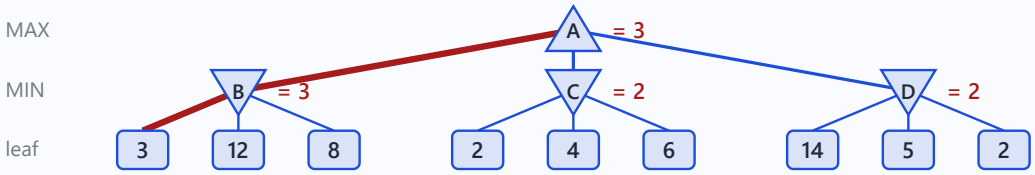


Figure 2.19: The result of the minimax tree. The MIN nodes take the values B = 3, C = 2, D = 2; the root A (MAX) chooses B and the value of the game is 3.

The notable point in the trace is node D. D's first leaf is 14, which looks attractive to MAX. But D is a MIN node: MIN chooses 2 instead of 14. Knowing this, MAX does not go to D. The value of the game is 3: MAX chooses B and, whatever MIN does, guarantees at least 3 (Figure 2.19). The code looks at all 9 leaves. Alpha-beta pruning can skip some leaves without evaluating them at all: for example, once C's first leaf 2 has been seen, C's value will be at most 2, which is below 3; there is no need to look at C's other leaves.

Common mistake

- **Looking for the largest leaf.** MAX cannot reach the largest number in the tree (14), because MIN makes the choice in between. Values are computed from the bottom up, at each node from the point of view of the player to move.
- **Mixing up the levels.** In these turn-taking trees, at the children of a node it is the other player's turn. In the code this is ensured by `not maxi` in the recursive call.

2.7.3 Tic-tac-toe: the whole tree can be searched

The tree of tic-tac-toe is small. A program that counts all the games finds 255,168 distinct games and 5,478 reachable positions. These numbers are tiny for a computer: the whole tree is searched in less than a second. The minimax values are taken as +1 (X wins), 0 (draw), and −1 (O wins). On the empty board the minimax value is 0: if both sides play without error, the game ends in a draw.

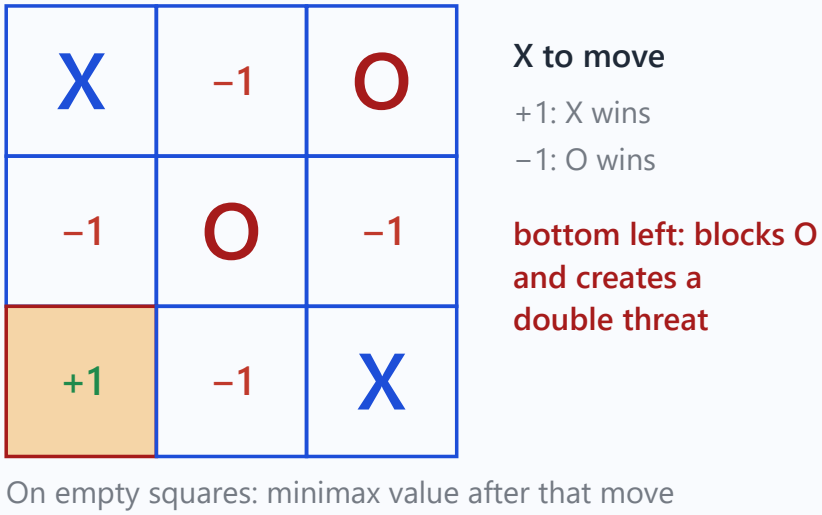


Figure 2.20: X to move. On each empty square the minimax value of the position is written for X playing there: +1 means X wins, −1 means O wins. Only the bottom-left square wins.

In the position of Figure 2.20, X wins only with the bottom-left square. This move does two things: it blocks O's diagonal (top right – center – bottom left) and at the same time creates two threats. In the left column (X, empty, X) and in the bottom row (X, empty, X) two separate winning threats arise. O can block only one; X completes the other (+1). On every other square, O completes their own diagonal and wins (−1).

In chess a full search is impossible; the tree has about 10^{120} games. Chess programs work with three techniques: a **depth limit** (the tree is cut after a certain number of moves), an **evaluation function** (the position at the cutoff is given a score: piece values, king safety, and so on), and **alpha-beta pruning** (branches that cannot change the result are skipped). Together these three techniques turn minimax into a strong player even in games where full search is impossible. A minimax tic-tac-toe opponent, in turn, plays by computing the values exactly: it cannot be beaten; the best one can hope for is a draw.

2.7.4 Code trace: Don't Say 21

The rule of Don't Say 21 is as follows: two players take turns, starting from 1, saying 1, 2, or 3 consecutive numbers. For example, the first player says "1, 2", the second says "3, 4, 5", the first says "6". Whoever says 21 loses. The state of the game is described by the last number said, c; at the start $c = 0$.

Each state is labeled W (the player to move **w**ins) or L (the player to move **l**oses). The base state is $c = 20$: the player to move must say 21 and loses (L). The other states are computed from the end backward: if from a state there is a move that pushes the opponent into an L state, the state is W; otherwise it is L.

```
TARGET = 21
wins = {TARGET - 1: False}   # c=20: lost
for c in range(TARGET - 2, -1, -1):
    wins[c] = any(not wins[c + k]
        for k in (1, 2, 3) if c + k < TARGET)
keys = [c for c in wins if not wins[c]]
print(sorted(keys))
```

The code works as follows:

- The dictionary `wins` holds, for each *c*, the answer to the question "does the player to move win?" `wins[20] = False`: the base state.
- The loop runs backward over *c* from 19 down to 0. For each *c*, saying 1, 2, or 3 numbers leads to the state *c* + *k* (without reaching 21; `c + k < TARGET`).
- `any(not wins[c + k] …)`: if at least one move leads to a state that loses for the opponent, *c* is a winning state.
- The list `keys` collects the losing states. These are the numbers a player wants to leave to the opponent: the **key numbers**.

c	States reachable	Can push the opponent into L?	Label
20	— (must say 21)	no	L
19, 18, 17	can reach 20	yes, says 20	W
16	**17, 18, 19 (all W)**	**no**	**L**
15, 14, 13	can reach 16	yes, says 16	W
12	**13, 14, 15 (all W)**	**no**	**L**
11, 10, 9	can reach 12	yes	W
8	**9, 10, 11 (all W)**	**no**	**L**
7, 6, 5	can reach 8	yes	W
4	**5, 6, 7 (all W)**	**no**	**L**
3, 2, 1	can reach 4	yes	W
0	**1, 2, 3 (all W)**	**no**	**L**

Table 2.17: Backward labeling in Don't Say 21.

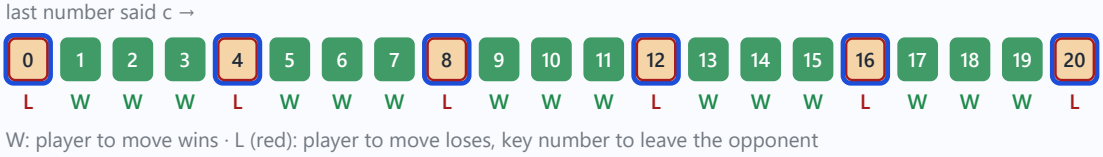


Figure 2.21: Labels of the states 0–20 in Don't Say 21. W: the player to move wins; L (highlighted): the player to move loses. The L states are the multiples of 4 and are the key numbers to leave to the opponent.

The code prints `[0, 4, 8, 12, 16, 20]`. The L states are the multiples of 4 (Figure 2.21). The starting state $c = 0$ is L: the first player loses. The second player's strategy is simple: if the opponent says x numbers, they say $4 - x$ numbers. In this way the total grows by 4 at the end of each round, and the second player always completes to 4, 8, 12, 16, 20. In the end the first player is forced to say 21.

Example 2.25

In "Don't Say 16" the rules are the same, but whoever says 16 loses. The key numbers are found by the same method:

1. Base state: $c = 15$ is L; the player to move must say 16.
2. 14, 13, 12: 15 can be reached → W. $c = 11$: 12, 13, 14 are all W → L.
3. The pattern continues in steps of 4: the L states are 15, 11, 7, 3.
4. $c = 0$: one can say 1, 2, or 3, and 3 is an L state. The starting state is W.

The first player wins: they say "1, 2, 3" and leave the opponent at 3. Then each time they complete to a multiple of 4 plus 3 (7, 11, 15). When the target number changes, the key numbers shift and the winning player can change. The general rule: if whoever says H loses, the L states are the numbers going back from $H - 1$ in steps of 4.

2.7.5 Nim and Bouton's theorem

In Nim there are several heaps of tokens. The player to move takes at least one token from a single heap (all of it, if they wish). Whoever takes the last token wins. Small positions can be labeled by hand as in Figure 2.18; but as the number and size of heaps grow, the number of positions increases rapidly. In 1901 Charles Bouton found a rule that labels all Nim positions with a single computation.

Definition 2.16

The bitwise XOR of the heap sizes is called the **nim-sum** and is written with $\oplus$. XOR is binary addition without carries: in each bit column the result is 1 if the number of 1s is odd and 0 if it is even.

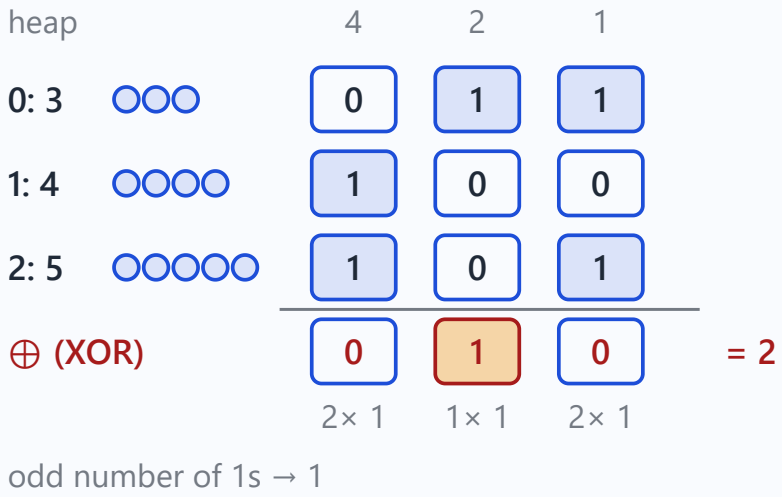


Figure 2.22: The nim-sum in the position [3, 4, 5]. The heaps are written in binary (3 = 011, 4 = 100, 5 = 101); the 1s are counted in each column. Only the 2s column has an odd number of 1s, so $3 \oplus 4 \oplus 5 = 010 = 2$.

Example 2.26

$3 \oplus 4 \oplus 5$ is computed column by column (Figure 2.22):

1. Binary notation: 3 = 011, 4 = 100, 5 = 101 (columns of 4s, 2s, 1s).
2. The 4s column: 0, 1, 1 → two 1s, even → 0.
3. The 2s column: 1, 0, 0 → one 1, odd → 1.
4. The 1s column: 1, 0, 1 → two 1s, even → 0.
5. The result is 010 = 2.

Theorem 2.2 (Bouton)

A Nim position is a losing position (P) if and only if its nim-sum is 0.

Proof. We check the two rules for P- and N-positions (Definition 2.15). The final position (all heaps empty) has nim-sum 0 and is P; this gives the base case.

(1) Every move from a position with nim-sum 0 goes to a position whose nim-sum is not 0. A move changes a single heap: it reduces its size h to a smaller h'. Since $h' \neq h$, the two differ in at least one bit. In that bit column the number of 1s increases or decreases by one, that is, its parity changes. In the previous sum that bit was 0; now it becomes 1. The sum is no longer 0.

(2) From a position with nim-sum $x \neq 0$ there is a move to a position with nim-sum 0. Take the leftmost 1 bit of x. The number of 1s in that bit column is odd, so at least one heap of size h has that bit equal to 1. Reduce this heap to $h \oplus x$. $h \oplus x$ is smaller than h: that bit turns from 1 to 0 and the bits further left do not change (because the bits of x to the left of that bit are 0). The move is legal. The new sum is $x \oplus h \oplus (h \oplus x) = 0$, because any number XORed with itself gives 0.

The positions with sum 0 satisfy the two rules of P, and those with nonzero sum satisfy the two rules of N. Since the labels are determined uniquely from the end backward, the P-positions are exactly those with nim-sum 0. □

The theorem also gives a strategy. In a position whose nim-sum is not 0, the player to move makes the move that makes the sum 0. With each of their moves the opponent, from a position with sum 0, breaks the sum; the player makes it 0 again each time. When all heaps are empty the sum is 0, and it is the player who leaves this position to the opponent, that is, who takes the last token. Many impartial games, in which both players have the same moves, can also be solved by converting them to Nim; the Sprague–Grundy theory is the general way of making this conversion.

2.7.6 Code trace: the winning move in Nim

The code below computes the nim-sum in a Nim position and finds the move that makes the sum 0.

```
heaps = [3, 4, 5]
x = 0
for h in heaps:
    x ^= h            # nim-sum (XOR)
for i, h in enumerate(heaps):
    if h ^ x < h:     # can it shrink?
        heaps[i] = h ^ x
        print("heap", i, ":", h, "->", h ^ x)
        break
print(heaps)
```

The first loop starts `x` at 0 and adds each heap with `^=` (XOR assignment). The second loop tries the heaps in order: if `h ^ x < h`, the heap can be reduced to `h ^ x` and this move makes the sum 0. When the first suitable heap is found, the move is made and the loop stops with `break`.

Step	Operation	Binary	Result
1	x = 0 XOR 3	000 ⊕ 011 = 011	x = 3
2	x = 3 XOR 4	011 ⊕ 100 = 111	x = 7
3	x = 7 XOR 5	111 ⊕ 101 = 010	x = 2
4	heap 0: 3 XOR 2	011 ⊕ 010 = 001	1 < 3 → suitable
5	heap 0: 3 → 1	—	[1, 4, 5], sum 0

Table 2.18: Trace for `heaps = [3, 4, 5]`.

The code prints "heap 0 : 3 -> 1" and `[1, 4, 5]`: two tokens are taken from the first heap. The choice of this heap is no accident. The leftmost 1 bit of `x = 2` is the 2s bit, and 3 (011) also has this bit equal to 1. What happens if the other heaps are tried? *4 ⊕ 2 = 6* and *5 ⊕ 2 = 7* are larger than the heaps; since adding tokens is forbidden, these moves cannot be made. In the new position [1, 4, 5], *1 ⊕ 4 ⊕ 5 = 0* (Figure 2.23).



Figure 2.23: [1, 4, 5] after the move. The number of 1s in each bit column is even and the nim-sum is 0; the opponent is in a losing position.

Example 2.27

Let the position be [2, 5, 6], with us to move.

1. Nim-sum: *2 ⊕ 5 = 010 ⊕ 101 = 111 = 7*; *7 ⊕ 6 = 111 ⊕ 110 = 001 = 1*. The sum is 1 ≠ 0: it is a winning position.
2. The heaps are tried: *2 ⊕ 1 = 3* (larger than 2, not allowed); *5 ⊕ 1 = 4* (smaller than 5, allowed); *6 ⊕ 1 = 7* (larger than 6, not allowed).
3. The only winning move: reduce the second heap from 5 to 4. The new position [2, 4, 6]: *2 ⊕ 4 ⊕ 6 = 010 ⊕ 100 ⊕ 110 = 000*.

The only 1 bit of *x = 1* is the 1s bit; the only heap with this bit equal to 1 is 5 (101). So there is exactly one winning move.

2.7.7 Solved games

Apart from the 2/3 game and the Prisoner's Dilemma, the interactive games of this chapter are two-player, sequential, perfect-information games without chance. By Zermelo's theorem all of them can in principle be solved by backward induction; but the tree is often very large. To "solve" a game means to find the outcome under perfect play (first player wins, second player wins, or draw). Table 2.19 summarizes the known results.

Game	Outcome under perfect play	How?
Tic-tac-toe	Draw	The whole tree is searched (minimax)
Nim	The player to move wins if the nim-sum ≠ 0	Bouton's XOR rule (1901)
Don't Say 21	The second player wins	Backward labeling; key numbers are multiples of 4

Game	Outcome under perfect play	How?
Connect Four	The first player wins	Solved by computer search in 1988
Hex	The first player wins (no draws)	Strategy stealing; the winning strategy is unknown on large boards
Othello (8×8)	Draw	By computer search in 2023
Checkers	Draw	Solved in 2007 (Schaeffer et al., 2007)
Chess	Determinate but unknown	Zermelo; tree $\approx 10^{120}$ games

Table 2.19: Some combinatorial games and their solution status.

In games such as Mancala, looking a few steps ahead in the move tree gives a player a significant advantage. In Fox and Geese the roles are asymmetric: on one side there is a single fox, on the other many geese; even so, information is perfect. Nine Men's Morris is also a classic strategy game of perfect information. The same principle holds in all of these games: if the game is finite and information is perfect, the outcome is in theory determined; the practical difficulty is the size of the tree.

2.8 Sequential games and backward induction

The "from the end backward" idea used in combinatorial games also applies to sequential games whose payoffs are not zero-sum. In these games a new problem arises: some Nash equilibria rest on threats that will never be carried out. The concept of subgame perfect equilibrium solves this problem, and backward induction is the algorithm that finds this equilibrium.

2.8.1 Subgames and subgame perfect equilibrium

Definition 2.17

A **subgame** is the part of a game tree that starts at a single decision node, contains everything below that node, and does not split any information set.

A **subgame perfect equilibrium** (SPE) is a strategy profile that forms a Nash equilibrium in every subgame (Selten, 1965).

The condition "does not split an information set" in the definition of a subgame says that the part starting at a node must not disturb what the player knows at that point. If one node of an information set lies inside the part and another outside, the part is not a subgame. In games of perfect information every information set is a single node, so every decision node starts a subgame. The whole game is also a subgame of itself.

Why is a new equilibrium concept needed? The ordinary Nash equilibrium does not look at what happens at nodes of the game that are never reached. A player can frighten the other with a threat they would never actually carry out; since the threat is never tested, the Nash equilibrium cannot catch it. Such a threat is called a **non-credible threat**. SPE requires players to make the best choice in every subgame, that is, at every node including those not reached. For this reason it rules out non-credible threats.

In finite games of perfect information the SPE is found by **backward induction** (Kuhn/Zermelo). The algorithm consists of four steps:

1. Find the last decision nodes, those connected only to leaves.
2. Choose the best branch of the player who decides there: only that player's payoff is compared.
3. Replace the node with the payoff vector of the chosen leaf; the node is now a leaf.
4. Repeat these steps until the root is reached.

The algorithm processes each node exactly once; its running time is therefore proportional to the size of the tree. If the tree is small the solution is fast. The difficulty in chess lies not in the algorithm but in the size of the tree.

2.8.2 Market entry: the non-credible threat

The market entry game (Figure 2.9) is written here with short names: E (the Entrant) chooses first; I (the Incumbent) decides only if E enters.

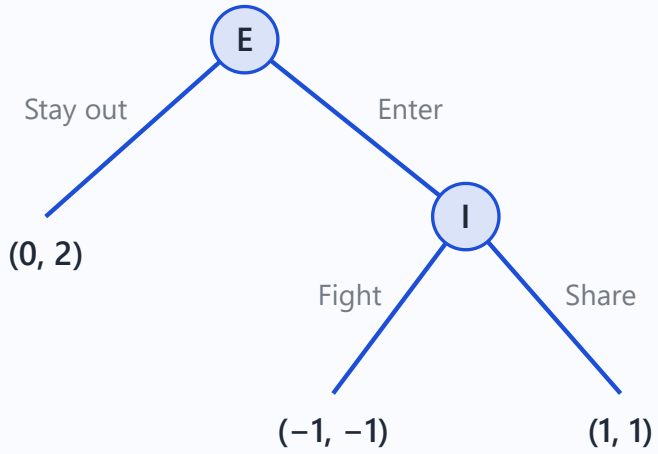


Figure 2.24: The market entry game. If E stays out, $(0, 2)$; if E enters, I fights $(-1, -1)$ or shares $(1, 1)$. The leaves show (u_E, u_I).

First the normal form of the game is written. E has two strategies: Enter, Stay out. I has a single decision node and two strategies: Fight, Share (that is, "if E enters I fight" and "if E enters I share"). If E stays out, I's strategy does not affect the outcome.

E \ I	Fight	Share
Enter	(−1, −1)	**(1, 1)**
Stay out	**(0, 2)**	(0, 2)

Table 2.20: The normal form of the market entry game. The green cells are the pure Nash equilibria.

Example 2.28

The best responses are marked in Table 2.20:

1. If I says Fight, Enter gives E −1 and Stay out gives 0 → Stay out. If I says Share, Enter gives E 1 and Stay out gives 0 → Enter.
2. If E says Enter, Fight gives I −1 and Share gives 1 → Share. If E says Stay out, both of I's strategies give 2 → both are best responses.
3. The cells with two marks: (Enter, Share) and (Stay out, Fight).

The game has two pure Nash equilibria. In the second, I says "if you enter, I will fight", and E, believing this threat, stays out. The threat is never tested; I's payoff is 2 and deviating gains nothing.

Backward induction rules out one of these two equilibria:

1. **Last node, I:** Fight → −1, Share → 1. I chooses Share. The value of I's node is (1, 1).
2. **Root, E:** Stay out → 0, Enter → 1 (knowing that I will share). E chooses Enter.
3. **SPE:** (Enter, Share); the outcome is (1, 1).

(Stay out, Fight) is a Nash equilibrium, but it is not subgame perfect. In the subgame starting at I's node, that is, after E has entered, fighting gives I −1 and sharing gives 1. In this subgame Fight is not a best response. The threat "if you enter, I will fight" is not credible: the moment E enters, I's interest is to share. A rational E foresees this and enters, ignoring the threat.

2.8.3 The investment game step by step

A three-stage game shows how backward induction works over several steps. Ayşe (A) invests or does not. If she invests, the entrepreneur Bora (B) either shares the profit or runs off with the money. If Bora runs off, Ayşe either sues or drops the matter. The leaves show the payoffs (Ayşe, Bora): Don't invest (2, 0); Share (3, 3); Run and Sue (1, 1); Run and Drop (0, 4).

Example 2.29

1. **Last decision node** (Ayşe, after Bora has run off): Sue 1, Drop 0. Since 1 > 0, Ayşe sues. The value of the node is (1, 1).
2. **Bora's node:** Bora knows that if he runs off he will be sued. Share 3, Run 1 (from the value of the lawsuit node). Since 3 > 1, Bora shares. The value of the node is (3, 3).
3. **Root** (Ayşe): Invest 3, Don't invest 2. Since 3 > 2, Ayşe invests.

Subgame perfect equilibrium: Ayşe's strategy is (Invest, Sue), Bora's strategy is Share; the outcome is (3, 3). Ayşe's strategy also contains the "Sue" component, because a strategy also says what to do at nodes that are not reached. The reason Bora shares is exactly this unreached choice.

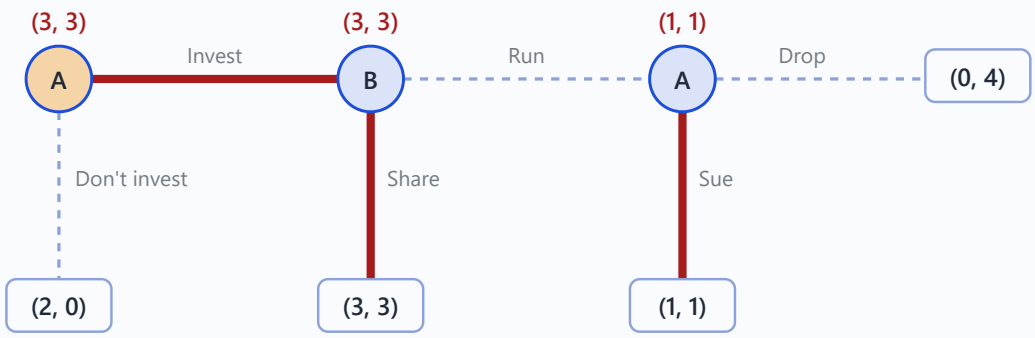


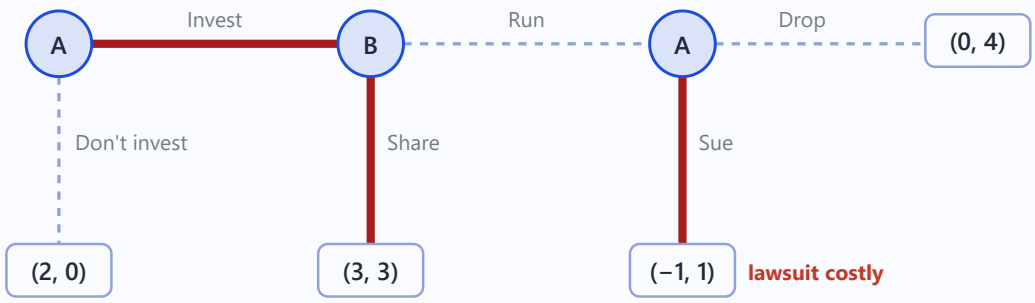
Figure 2.25: The solution of the investment game. The bold red branches are the action chosen at each node; the values found by backward induction are written above the nodes. The outcome is (3, 3).

Now let the lawsuit be expensive: Ayşe, if she sues, gets −1 instead of 1. The tree is the same; only one leaf has changed, from (1, 1) to (−1, 1). Backward induction is carried out again from the start.

Example 2.30

1. **Last node:** Drop 0, Sue −1. Since 0 > −1, Ayşe drops. The value of the node is (0, 4).
2. **Bora:** Run 4, Share 3. Since 4 > 3, Bora runs. The value of the node is (0, 4).
3. **Root:** Invest 0, Don't invest 2. Since 2 > 0, Ayşe does not invest.

The outcome (2, 0) is worse for both than (3, 3). The change of a single leaf has, in a chain, changed the decision at every node.



A: Ayşe (investor), B: Bora (entrepreneur); leaves show (Ayşe, Bora)

Figure 2.26: When the lawsuit is expensive (leaf (−1, 1)), the threat "if you run, I will sue" is not credible. The path (Invest, Share) can still be supported by a Nash equilibrium, but it is not subgame perfect.

Figure 2.26 points to a subtle matter. If Ayşe announces the plan "if you run off, I will sue", the profile (Ayşe: Invest, Sue; Bora: Share) is still a Nash equilibrium. In this profile Ayşe gets 3 and Bora gets 3. If Bora runs off he is sued and gets 1; he does not deviate. If Ayşe does not invest she gets 2; she does not deviate. Since the lawsuit node is never reached, Ayşe's choice there does not affect her payoff. But the threat is not credible: if Bora really runs off, Ayşe gets −1 by suing and 0 by dropping, and she drops. Bora knows this. This profile is not a subgame perfect equilibrium.

The lesson here is that making promises is not enough; carrying out the threat must be profitable at that moment. There are ways for Ayşe to make suing profitable: agreeing in advance with a lawyer on a fixed fee, or taking collateral from Bora. Measures of this kind change the payoffs of the game and make the threat credible.

2.8.4 Code trace: solve()

Backward induction can be written as a short recursive function. The tree is represented with nested lists: each decision node has the form `[name, player, branches]`; `branches` is a list of `(move, child)` pairs. The leaves are payoff tuples (`tuple`). Players are indexed 0 and 1: player 1 is index 0, player 2 is index 1.

```
def solve(d):
    if isinstance(d, tuple):         # leaf: payoffs
        return d
    name, player, branches = d
    best = None
    for move, child in branches:
        v = solve(child)
        if best is None or v[player] > best[player]:
            best, choice = v, move
    print(name, choice)
    return best
T = ['root', 0, [('Left', ['x', 1, [('a', (3, 1)), ('b', (1, 2))]]),
                 ('Right', ['y', 1, [('c', (2, 1)), ('d', (0, 0))]])]]
print(solve(T))
```

The function works as follows:

- If the node is a tuple it is a leaf, and the payoff vector is returned as is.
- Otherwise the node is unpacked into three parts: `name`, the deciding `player`, and `branches`.
- For each branch the value of the child is computed recursively: `v = solve(child)`.
- In the comparison only *the deciding player's* payoff is used: `v[player] > best[player]`. Because strict greater-than (`>`) is used, the first branch seen is kept in a tie.
- The node's choice is printed and the payoff vector of the chosen branch is returned; the parent node uses this vector like a leaf.

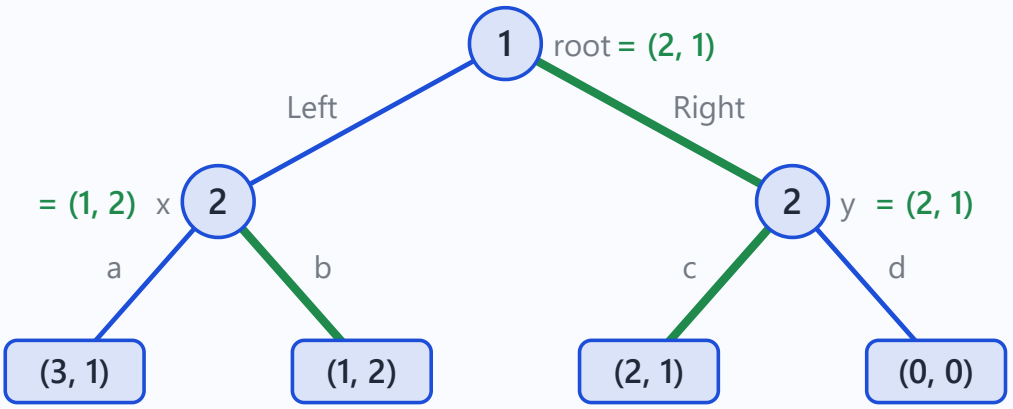


Figure 2.27: The tree of `solve(T)` and its result. Player 1 chooses at the root, player 2 at nodes x and y; b is chosen at x, c at y, and Right at the root. The outcome is (2, 1).

Step	Node (player)	Branch and value v	Comparison	best, choice
1	x (1)	a → (3, 1)	no best yet	(3, 1), a
2	x (1)	b → (1, 2)	v[1] = 2 > 1	(1, 2), b → "x b"

Step	Node (player)	Branch and value v	Comparison	best, choice
3	root (0)	Left → (1, 2)	no best yet	(1, 2), Left
4	y (1)	c → (2, 1)	no best yet	(2, 1), c
5	y (1)	d → (0, 0)	v[1] = 0 > 1? no	(2, 1), c → "y c"
6	root (0)	Right → (2, 1)	v[0] = 2 > 1	(2, 1), Right → "root Right"

Table 2.21: Trace of `solve(T)`.

The program prints "x b", "y c", "root Right", and `(2, 1)` in turn. Step 2 is important: at x player 2 looks at their own payoff and chooses b (2 > 1). In this way player 1's favorite leaf (3, 1) becomes unreachable. At the root, player 1 compares the true value of Left, (1, 2), with the true value of Right, (2, 1), and chooses Right. The SPE is (Right; b at x, c at y). The b at the unreached node x is also part of the strategy; it is the reason player 1 rejects Left.

2.8.5 The Centipede game

The Centipede game was described by Robert Rosenthal in 1981 (Rosenthal, 1981). Two players decide in turn. At each node the deciding player either says "Take" and the current split is carried out, or says "Pass" and the total grows. The total payoff grows as 1 → 2 → 4 → 6 and is 6 at the end. But a player who passes receives less if the opponent takes at the next step.

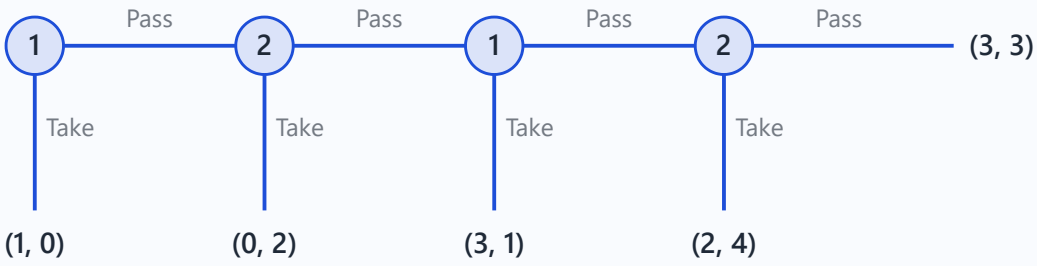


Figure 2.28: A Centipede game with four nodes. The number in a node is the deciding player. The downward branches are Take; the leaves show the payoffs (Player 1, Player 2). If everyone passes, the outcome is (3, 3).

Reading the leaves in order shows the tension of the game. If player 1 takes at the first node, the outcome is (1, 0). If they pass, player 2 can take at the second node and get (0, 2): for player 2, 2 > 0. If player 2 passes, player 1 can take at the third node with (3, 1). At the fourth node player 2 can take with (2, 4) or pass and go to (3, 3).

Backward induction starts from the last node. Table 2.22 shows the result of the four steps.

Node	Decides	Take	Value of passing	Decision	Node value
4	Player 2	(2, 4)	(3, 3)	Take	(2, 4)
3	Player 1	(3, 1)	(2, 4)	Take	(3, 1)

Node	Decides	Take	Value of passing	Decision	Node value
2	Player 2	(0, 2)	(3, 1)	Take	(0, 2)
1	Player 1	(1, 0)	(0, 2)	Take	(1, 0)

Table 2.22: Backward induction in the Centipede game.

Example 2.31

1. **Node 4, Player 2:** Take 4, Pass 3 (the value of passing is (3, 3)). 4 > 3 ⇒ Take; the node's value is (2, 4).
2. **Node 3, Player 1:** Take 3, Pass 2 (the value of passing is (2, 4)). 3 > 2 ⇒ Take; the node's value is (3, 1).
3. **Node 2, Player 2:** Take 2, Pass 1 (the value of passing is (3, 1)). 2 > 1 ⇒ Take; the node's value is (0, 2).
4. **Node 1, Player 1:** Take 1, Pass 0 (the value of passing is (0, 2)). 1 > 0 ⇒ Take; the node's value is (1, 0).

Subgame perfect equilibrium: each player says Take at every node. The game ends at the first move with (1, 0), although (3, 3) was better for both.

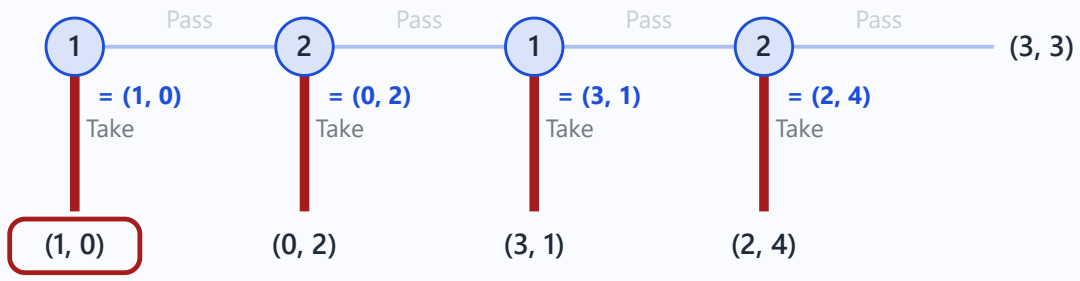


Figure 2.29: The solution of the Centipede game. Take (bold red) is chosen at every node; the node values are found from right to left as (2, 4), (3, 1), (0, 2), (1, 0). The game ends at the first node with (1, 0).

Example 2.32

Let the rightmost payoff be (5, 5) instead of (3, 3). Backward induction is carried out again:

1. Node 4, Player 2: Pass 5 > Take 4 ⇒ Pass; value (5, 5).
2. Node 3, Player 1: Pass 5 > Take 3 ⇒ Pass; value (5, 5).
3. Node 2, Player 2: Pass 5 > Take 2 ⇒ Pass; value (5, 5).
4. Node 1, Player 1: Pass 5 > Take 1 ⇒ Pass; value (5, 5).

Everyone passes and the outcome is (5, 5). The key to solving the game is the last node: the moment player 2 there prefers not to take, the whole chain reverses.

2.8.6 Centipede: theory and experiment

Theory. In the SPE each player says Take at every node and the game ends at the first move with (1, 0). There is also a stronger result: in any Nash equilibrium of the game the outcome is the same. The reason is this. If there were a Nash equilibrium in which the game ended after node 1, the player deciding at the node just before where the game ends would want to deviate. For example, if the game ended at node 3 with (3, 1), player 2 at node 2 could have taken and got 2, while passing gives 1. If the game ran to the end with (3, 3), player 2 at node 4 could take and get 4. In every case there is a profitable unilateral deviation; so in every Nash equilibrium the game ends at the first node.

Inefficiency. (3, 3) is better than (1, 0) for both players: the SPE outcome is not Pareto optimal. The loss grows with the number of nodes; in a centipede with a hundred nodes the players sacrifice a large total at the first step.

Experiment. In the laboratory most players do not take at the first node; they pass for a few rounds (McKelvey and Palfrey, 1992). The reason is that backward induction requires rationality to be common knowledge and a long chain of reasoning. Even a small chance that the opponent is somewhat "well-intentioned" makes passing profitable: if the opponent passes one more round, the player can take from a larger pot.

The Centipede game shares its logic with the finitely repeated Prisoner's Dilemma: looking backward from the last round, cooperation unravels. With an infinite horizon, however, cooperation can remain in equilibrium (Section 2.8.8).

2.8.7 The Ultimatum game

The Ultimatum game is the simplest model of bargaining (Güth, Schmittberger, and Schwarze, 1982). Ayşe offers Bora x TL out of 10 TL; x is a whole number of TL (0, 1, …, 10). If Bora accepts, the split is $(10 - x, x)$. If he rejects, both get 0.

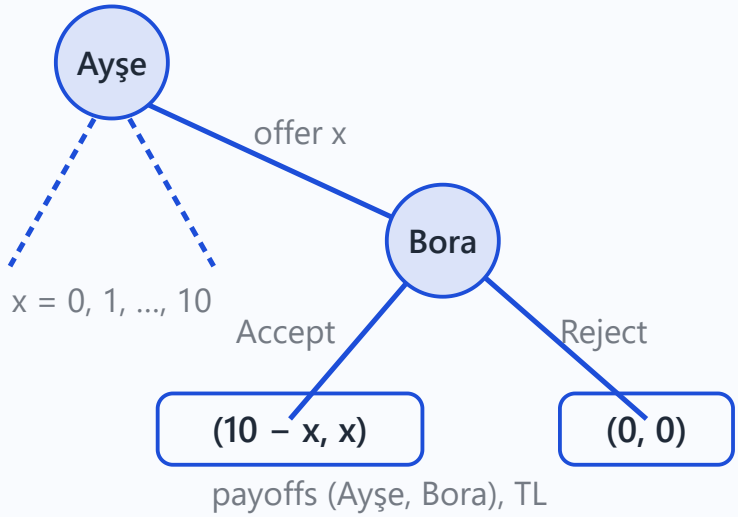


Figure 2.30: The tree of the Ultimatum game. Ayşe offers $x = 0, 1, \ldots, 10$; for each offer Bora accepts or rejects. Payoffs (Ayşe, Bora), in TL.

Example 2.33

1. **Bora's nodes.** There is a separate node for each x (11 nodes). If $x > 0$, accepting gives x and rejecting gives 0; since $x > 0$, Bora accepts. At $x = 0$ both accepting and rejecting give 0; Bora is indifferent.
2. **Ayşe's decision (if Bora rejects 0).** Ayşe knows that Bora will accept every positive offer. The positive offer that maximizes her share $10 - x$ is $x = 1$. The SPE outcome is (9, 1).
3. **Ayşe's decision (if Bora accepts 0 too).** Ayşe offers $x = 0$ and gets 10. This is also an SPE; the outcome is (10, 0).

Bora's strategy contains a decision at each of the 11 nodes. Bora therefore has $2^{11} = 2048$ strategies.

There is a non-credible threat in this game too. If Bora says "I will reject anything below 4 TL" and Ayşe believes it, Ayşe offers 4. The profile (Ayşe: offer 4; Bora: accept 4 and above, reject below) is a Nash equilibrium: if Ayşe offers less she gets 0, if she offers more she keeps less; Bora's choice at the node actually reached (accept) is best. But when 1 TL is offered, rejecting gives Bora 0 < 1; the threat is not credible and the profile is not subgame perfect.

Offer (share of the total)	Theory (money only)	In experiments
Most common offer	the smallest unit	40–50%
Offers below 20%	accepted	about half are rejected

Table 2.23: Theory and experiment in the Ultimatum game (Camerer, 2003).

The experimental results are very different from the theory. Most proposers offer 40–50% of the total; about half of the offers below 20% are rejected. This result should not be read as "players cannot do backward induction". Rejection shows that a concern for fairness enters the utility function: rejecting an unfair offer may be worth more to someone than 1 TL. Proposers foresee this and behave generously (Camerer, 2003). This is an example of the principle in Section 2.1.5: rational does not mean selfish.

2.8.8 The infinitely repeated Prisoner's Dilemma

Is cooperation possible if the Prisoner's Dilemma is played repeatedly between the same two players? The answer depends on whether it is known when the game will end. The same one-shot game is repeated; what determines the outcome is the horizon.

Definition 2.18

Grim trigger strategy: Play Silent in the first round; if the opponent betrays even once, play Betray forever after.

Finite, known horizon. Let the game last 10 rounds, and let this be known. In the last round there is no future and hence no punishment: round 10 is a one-shot Prisoner's Dilemma, and there Betray is dominant. In round 9 the players know what will happen in round 10: whatever they do, both will betray in round 10. So the choice in round 9 has no future either, and round 9 too is like a one-shot game: Betray. Backward induction carries this all the way to the first round. Betray is played in every round; as in the Centipede game, cooperation unravels from the end backward.

Infinite horizon. After each round let the game continue for one more round with probability δ. The game has no known last round. The gain from today's betrayal is weighed against the loss from the permanent punishment in the future. If δ is large enough, that is, if the players are patient or the probability that the game continues is high, deviating is not profitable and cooperation remains in equilibrium.

Example 2.34

With Sam and John's payoffs (Table 2.8), let both players play grim trigger. We compute whether a deviation by Sam is profitable.

1. **Complying.** (Silent, Silent) is played every round and Sam gets −1 every round. Total: $-1 - \delta - \delta^2 - \ldots = -1 / (1 - \delta)$ (by the formula of Example 2.11).
2. **Deviating.** Sam betrays in one round and gets 0 that round (John is silent). In every later round John betrays; Sam's best reply is also to betray, and he gets −2 every round. Total: $0 + \delta(-2) + \delta^2(-2) + \ldots = -2\delta / (1 - \delta)$.
3. **Comparison.** Complying is no worse than deviating if: $-1 / (1 - \delta) \geq -2\delta / (1 - \delta)$. Multiplying both sides by $(1 - \delta) > 0$: $-1 \geq -2\delta$, that is, $\delta \geq 1/2$.

If the probability that the game lasts one more round is at least 1/2, betrayal is unprofitable and two grim trigger strategies together sustain cooperation. For example, at $\delta = 0.9$ complying gives −10 and deviating gives −18.

The punishment is credible, because (Betray, Betray) is an equilibrium of the one-shot game. In the punishment phase no player can gain by staying silent alone: while the opponent betrays, staying silent gives −3 and betraying gives −2. So two grim trigger strategies together form a subgame perfect equilibrium: nobody wants to deviate either in the cooperation phase (when $\delta \geq 1/2$) or in the punishment phase.

2.8.9 Common mistakes in sequential games

Common mistake

- **Starting at the root.** The first move depends on the response to later moves; the solution always starts from the last decision nodes.
- **Looking at the wrong player's payoff.** At each node only the number of the player who decides there is compared. In the investment game, at Bora's node the second number is examined.
- **Mistaking a strategy for a path.** A strategy contains a choice for every node, including unreached ones: Ayşe's strategy is (Invest, Sue), not just "Invest".
- **Thinking Nash = subgame perfect.** Every subgame perfect equilibrium is a Nash equilibrium; the converse is false. There are Nash equilibria that contain non-credible threats (market entry, ultimatum).
- **Solving a simultaneous game as sequential.** If moves are not observed (Prisoner's Dilemma), the tree has an information set; backward induction does not apply directly, and the matrix is solved instead.
- **Mistaking a promise for commitment.** Saying "I will sue" is not enough; carrying out the threat must be profitable at that moment or be bound in advance (contract, collateral).

2.9 Minimax and alpha-beta pruning

The definition of the minimax value takes a few lines: at a leaf the outcome of the game, at a MAX node the largest of the children, at a MIN node the smallest. This definition is easily applied by hand on small trees and by computer on tic-tac-toe. The difficulty lies not in the definition but in the size of the tree. Looking one move further ahead multiplies the number of positions to examine by the branching factor. So the strength of game-playing programs depends on their answers to three questions: which branches can be skipped without being read at all, in what order should moves be

tried, and, if the tree cannot be descended to the leaves, what value should be given to the positions where it is cut short? Alpha-beta pruning answers the first question, move ordering the second, and the evaluation function with depth-limited search the third.

2.9.1 The game tree and its size

In the extensive form a game is written as a tree. In two-player, zero-sum games of perfect information only one player decides at each level of this tree, and a single number suffices to write the payoffs. The vocabulary used for such games is collected in the definition below.

Definition 2.19

In the **game tree** of a two-player, zero-sum game of perfect information, each **node** is a position: the state of the board and whose turn it is. Each **edge** is a move. The **root** is the current position; a **leaf** is a position where the game has ended and has no children. The player to move at the root is called **MAX** (▲) and the opponent **MIN** (▽). MAX wants to increase the value and MIN to decrease it. The levels alternate between MAX and MIN; going down one level is a **ply**, that is, a single move by one player.

Each leaf has a **utility**, and this number is written from MAX's point of view. Since the game is zero-sum, MIN's utility is its negative.

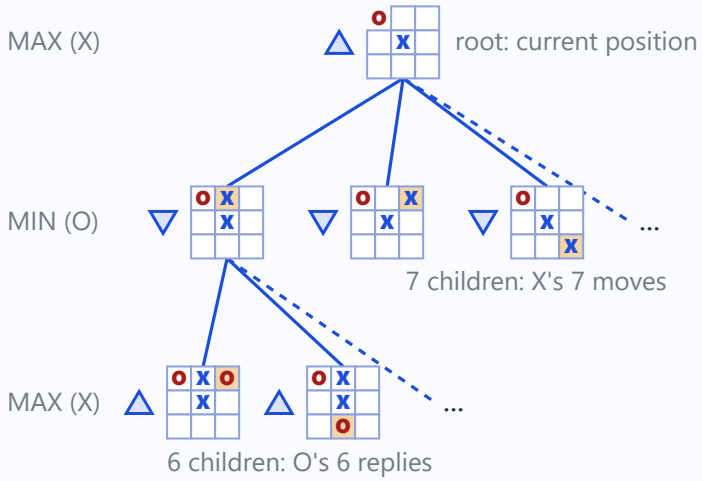


Figure 2.31: The first two levels of the tic-tac-toe game tree. Two stones have been placed at the root and it is X's turn (MAX). The seven children of the root are X's seven moves; below each are O's six replies (MIN).

Figure 2.31 shows how the tree grows. At the root there are 7 empty squares, so X has 7 moves. After each move 6 empty squares remain, so O has 6 replies. After only two plies there are $7 \cdot 6 = 42$ positions below the root. The same position can also be reached by different move orders: X playing

top-left first and bottom-right second gives the same board as bottom-right first and top-left second. In the tree this position appears as two separate nodes; the tree is a tree of game histories, not of positions.

In tic-tac-toe the utility takes three values: +1 if X wins, 0 for a draw, −1 if O wins (Figure 2.32). O's utility is the negative of X's: $u_O = -u_X$. So writing a single number at each leaf suffices. The values of the interior nodes are not known at the start; minimax computes them from the leaves. A judgment such as "the center is a good square" also becomes a number only once the tree is searched down to the leaves: the value of the center is the utility of the leaf reached by best play from both sides after playing in the center.



Figure 2.32: Three game endings and their utilities (from X's point of view): X wins +1, draw 0, O wins −1.

Two numbers determine the size of the tree.

Definition 2.20

The **branching factor** b is the number of legal moves in a position; since it changes during the game, its average or its maximum is often used. The **depth** d is the length of the game in plies. If every interior node has b children, there are b^d leaves at depth d; the size of the tree is roughly b^d.

This number is exponential: when the depth increases by one, the work is multiplied by b. In chess $b \approx 35$, so looking one more ply ahead multiplies the work by about 35, and looking one more full move (two plies) ahead multiplies it by about 1,225. Table 2.24 compares the trees of four games.

Game	b	d (plies)	Size of the tree or number of positions
Tic-tac-toe	≤ 9	≤ 9	9! = 362,880 upper bound; in reality 255,168 games, 549,946 nodes
Connect Four	≤ 7	≤ 42	4,531,985,219,092 ≈ $4.5 \cdot 10^{12}$ legal positions (Tromp)
Chess	≈ 35	≈ 80	$35^{80} \approx 10^{123}$; Shannon's (1950) rough estimate ≈ 10^{120} games
Go (19×19)	≈ 250	≈ 150	≈ $2.1 \cdot 10^{170}$ legal positions (Tromp, 2016)

Table 2.24: Branching factor, depth, and the number of tree nodes or positions in four games.

Example 2.35

The size of the tic-tac-toe tree can be expressed with four different numbers. The difference among the four lies in what is counted in the tree.

1. **Upper bound.** If no game ended early there would be 9 options on the first move, 8 on the second, …, 1 on the last: $9 \cdot 8 \cdot \ldots \cdot 1 = 9! = 362{,}880$ distinct games.
2. **The actual number of games.** Some games end early when a line of three is completed; those games have no continuation. A program that counts all the games finds 255,168 distinct games. Each game is a leaf of the tree.
3. **Number of nodes.** A full minimax search from the empty board visits not only the leaves but every interior node on the way. The number of nodes visited, including the root, is 549,946; of these 255,168 are leaves and the remaining 294,778 are interior nodes.
4. **Number of distinct positions.** Because the same board is reached by different orders, the 549,946 nodes in the tree correspond to only 5,478 distinct positions.

The roughly 100-fold difference between the last two numbers shows that the same positions are searched over and over. This repetition can be eliminated with a memory called a transposition table.

The whole tic-tac-toe tree is searched in under a second. Even in Connect Four, a full search has been possible only with game-specific knowledge and very strong programs. In chess and Go, searching the whole tree is out of the question; the search is stopped at a certain depth and the value of the position there is estimated.

2.9.2 The minimax value and a worked solution

The minimax value is the utility MAX will receive at the end of the game, assuming both sides play their best. The definition ties the value of a position to the values of its children; it is therefore recursive.

Definition 2.21

The **minimax value** of a position *s* is defined as follows:

```
minimax(s) =
  utility(s)                    if s is terminal
  max  minimax(result(s, a))    if MAX to move
   a
  min  minimax(result(s, a))    if MIN to move
   a
```

Here *a* ranges over the legal moves in *s*, and *result(s, a)* is the position after move *a*. At the root, MAX chooses the move that goes to the child with the largest value.

The search that applies the definition proceeds **depth-first**: it descends from the root to a leaf, reads the leaf's utility, returns to the node above, and descends into that node's other children. Values **rise** from the leaves to the root in this order. The value of a node becomes final only when all its children are finished.

The cost of the search is measured in two items. **Time:** since each node is visited once, $O(b^d)$. **Memory:** only the path from the root to the current node and, along this path, which children of each node have been tried are kept at any moment; this comes to $O(b \cdot d)$. Because memory is this small, the limit of minimax is never memory but time.

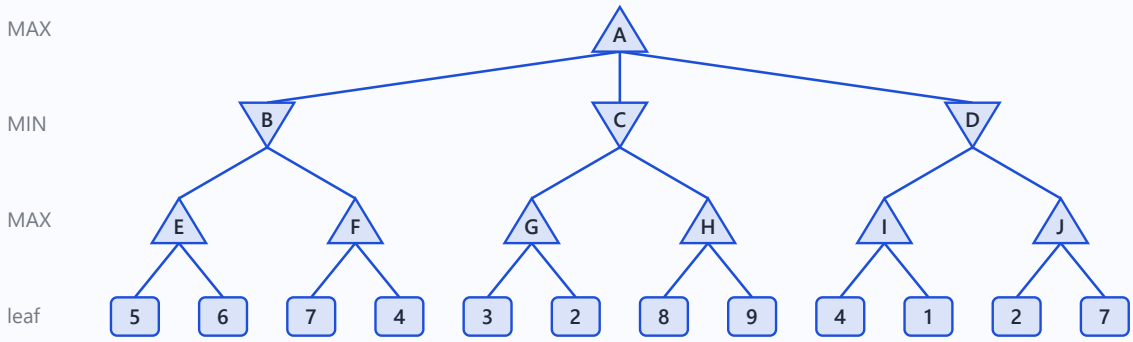


Figure 2.33: A three-level game tree: root A (MAX), B, C, D (MIN) at the second level, E–J (MAX) at the third level, and twelve leaves. The values of the interior nodes have not been computed yet.

Example 2.36

In the tree of Figure 2.33, the leaves from left to right are E = [5, 6], F = [7, 4], G = [3, 2], H = [8, 9], I = [4, 1], J = [2, 7]. The value of the root and MAX's move are found from the bottom up, level by level:

1. **Third level (MAX).** Each MAX node takes the largest of its leaves: E = max(5, 6) = 6, F = max(7, 4) = 7, G = max(3, 2) = 3, H = max(8, 9) = 9, I = max(4, 1) = 4, J = max(2, 7) = 7.
2. **Second level (MIN).** Each MIN node takes the smallest of its children: B = min(E, F) = min(6, 7) = 6, C = min(G, H) = min(3, 9) = 3, D = min(I, J) = min(4, 7) = 4.
3. **Root (MAX).** A = max(B, C, D) = max(6, 3, 4) = 6.

The value of the root is 6, and MAX chooses the move to B. The largest leaf in the tree (9) lies below H, but MAX cannot reach it: at C, MIN makes the choice and picks G, that is, 3. The 7 below B is also an unreachable number; at B, MIN chooses E instead of F and holds MAX to 6. The value 6 is the best outcome MAX can guarantee whatever the opponent does.

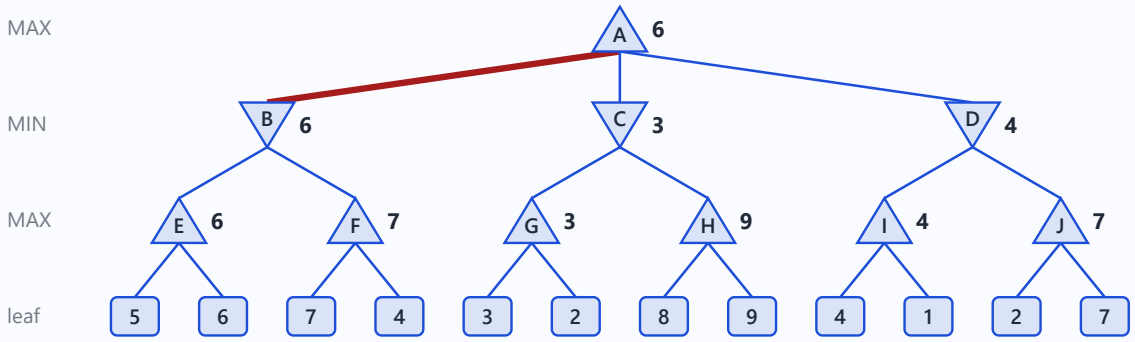


Figure 2.34: The result of minimax. The value of each interior node is written to its right; the bold edge is the move chosen at the root (A → B). The value of the root is 6.

Minimax is the special case of backward induction for two-player, zero-sum games. In a general sequential game, the player to move at each node maximizes *their own* payoff. In a zero-sum game the second player's payoff is the negative of the first's, so "the second player maximizes their own payoff" and "minimizes the first player's payoff" are the same thing. This is why the single number in the tree is maximized at MAX levels and minimized at MIN levels.

This also links minimax to the maximin and minimax criteria defined for matrix games (Example 2.7). Maximin is the best value MAX can guarantee alone: $\max_s \min_t u(s, t)$. Minimax is the lowest value to which MIN can hold MAX: $\min_t \max_s u(s, t)$. In simultaneous games such as Matching Pennies these two numbers can differ under pure strategies. In a finite game of perfect information they are always equal, and pure strategies suffice for the equality (Zermelo, 1913; Kuhn, 1953). The minimax value computed in the tree is this common value. The reason is that in perfect information each player sees the opponent's earlier moves when deciding: there is no "what did the opponent choose?" uncertainty of the simultaneous game.

The minimax move is found assuming the opponent plays without error. The guarantees and limits of this assumption are as follows:

- **What it guarantees:** MAX gets at least the minimax value whatever the opponent does. If the opponent errs, MAX can get more; never less. So the minimax move is the move that maximizes the worst-case payoff.
- **What it does not guarantee:** It does not find the move that earns the most against a weak opponent. It does not distinguish between two moves of equal value, one of which sets a trap for the opponent and the other does not; it sees both with the same number.
- **When there is chance:** In games with dice such as backgammon, **chance nodes** are added to the tree. At a chance node one takes neither the largest nor the smallest value but the probability-weighted average (*expectiminimax*). Pruning becomes harder in such trees, because to bound an average one has to see most of the children.

2.9.3 Code trace: the minimax function

The code below applies the definition to a smaller tree. The root A has three MIN children: B = [3, 5], C = [2, 9], D = [6, 4].

```
from math import inf
T = {"A": ["B", "C", "D"], "B": [3, 5],
     "C": [2, 9], "D": [6, 4]}
def minimax(n, maxi):
  if isinstance(n, int):        # leaf
    return n
  best = -inf if maxi else inf
  for c in T[n]:
    v = minimax(c, not maxi)
    if maxi:
      best = max(best, v)
    else:
      best = min(best, v)
  return best
print(minimax("A", True))
```

The structure of the code is as follows:

- The dictionary `T` holds the children of each interior node in order. The leaves are integers; whether a node is a leaf is decided by the test `isinstance(n, int)`, and the leaf's utility is returned at once.
- `best` starts at `-inf` at a MAX node and `inf` at a MIN node. The infinities mean "no child seen yet": whatever the value of the first child, it is larger than $-\infty$ for MAX and smaller than $+\infty$ for MIN, and replaces `best`.
- The loop tries the children in the order of `T`. At a child it is the other player's turn; this is the reason the recursive call is made with `not maxi`.
- A MAX node updates with `max`, a MIN node with `min`. When the loop ends, the node's final value is returned.

Step	Stack	Event	best
1	A	root A (MAX) is opened	−∞
2	A ▸ B	B (MIN) is opened	+∞
3	A ▸ B	leaf 3: min(+∞, 3)	3
4	A ▸ B	leaf 5: min(3, 5); B finishes, returns 3	3
5	A	B = 3: max(−∞, 3)	3
6	A ▸ C	C is opened; leaves 2 and 9: min(+∞, 2) = 2, min(2, 9) = 2	2
7	A	C = 2: max(3, 2)	3
8	A ▸ D	D is opened; leaves 6 and 4: min(+∞, 6) = 6, min(6, 4) = 4	4
9	A	D = 4: max(3, 4); A finishes	4

Table 2.25: Trace of `minimax("A", True)`. The stack is the calls open at that moment.

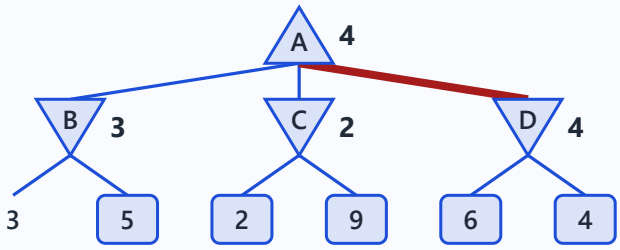


Figure 2.35: The end of the code trace. B = 3, C = 2, D = 4; the root A = 4 and A chooses the move to D. All six leaves have been read.

The program prints `4`. Two points in the trace are instructive. First, the root's `best` rises step by step from 3 to 4: A's value is not final until the last child D is finished. Second, reading C's second leaf, 9, is unnecessary. Once C's first leaf 2 has been seen, C's value will be at most 2, while A has already been guaranteed 3 from B. C will not be chosen whatever 9 is. Minimax does not know this and reads all six leaves. Alpha-beta pruning skips exactly this kind of reading.

2.9.4 Nim: minimax, P- and N-positions, and the XOR rule

In Nim, positions are labeled P and N (Definition 2.15): in an N-position the player to move wins, in a P-position the previous player wins. These labels are another way of writing the minimax value. To see this, write the value not from MAX's but from *the player to move's* point of view: +1 if the player to move wins, −1 if they lose. After a move the turn passes to the opponent, so the child's value is from the opponent's point of view and is read with its sign flipped:

```
value(s) = -1                              if there is no move
value(s) = max  -value(result(s, a))       otherwise
           a
```

This is the writing of minimax as a single function and is called **negamax**. Choosing the smallest at a MIN node is the same as choosing the largest of the negated values; negamax merges the two separate branches with this observation.

The formula repeats the P and N rules word for word. A position with no move is −1, that is, P. A position's value is +1 if and only if at least one child's value is −1; that is, if at least one move goes to a P-position the position is N. If all children are +1 the value is −1; that is, if every move goes to an N-position the position is P. Consequently the correspondence N ⇔ +1 and P ⇔ −1 holds.

Position	Value to the player to move	Type
(0, 0)	−1	P
(1, 0)	+1	N
(1, 1)	−1	P
(2, 0)	+1	N
(1, 2)	+1	N
(2, 2)	−1	P
(1, 2, 3)	−1	P

Table 2.26: Minimax value and type, for the player to move, of some Nim positions.

Example 2.37

The value of the position Nim(1, 3) is found with negamax. There are four moves from the position: taking the single token from the first heap, giving (0, 3), or taking 1, 2, or 3 tokens from the second heap: (1, 2), (1, 1), (1, 0).

1. (1, 1): both moves leave (0, 1) or (1, 0). From these one goes to (0, 0), and since (0, 0) = −1, (0, 1) = (1, 0) = +1. (1, 1) = max(−(+1), −(+1)) = −1: P.
2. (1, 2) = +1, because it can go to (1, 1): −(−1) = +1. (0, 3) = +1, because one can take all three tokens at once and leave (0, 0). (1, 0) = +1.
3. (1, 3) = max(−(0, 3), −(1, 2), −(1, 1), −(1, 0)) = max(−1, −1, +1, −1) = +1.

The position is N, and the only winning move is to take two tokens from the second heap, leaving (1, 1). The nim-sum is $1 \oplus 3 = 2$; the move that makes the sum 0 is $3 \oplus 2 = 1$, that is, reducing the second heap to 1. The two methods find the same move. The full tree contains 28 nodes.

By Bouton's theorem (Theorem 2.2), a Nim position is P if and only if its nim-sum is 0, and the winning move is the move that makes the sum 0. Minimax finds this without knowing any theory, by searching the whole tree. Table 2.27 compares the two methods on five positions.

Position	XOR	Minimax's winning move	Move making XOR 0	Full tree
(1, 2)	3	→ (1, 1)	→ (1, 1)	12 nodes
(1, 3)	2	→ (1, 1)	→ (1, 1)	28 nodes
(2, 3)	1	→ (2, 2)	→ (2, 2)	86 nodes
(1, 2, 3)	0	none: P-position	none	447 nodes
(3, 4, 5)	2	→ (1, 4, 5)	→ (1, 4, 5)	1,038,768 nodes

Table 2.27: The move minimax finds in the full tree versus the move of the XOR rule. The last column is the number of nodes of the full tree, including the root.

The two methods find the same move in every row; the difference is the cost. The tree of minimax grows exponentially with the number of tokens: a tree of 12 nodes for (1, 2) exceeds a million for (3, 4, 5), with 12 tokens in total. The XOR rule finishes with as many operations as there are heaps. When a game-specific theory is found there is no need to search; when there is no such theory, minimax is the general method that works on every game.

2.9.5 Value scales and the depth penalty in tic-tac-toe

In tic-tac-toe the +1 / 0 / −1 scale gives the outcome of the game correctly, but it has a flaw: a fast win and a slow win get the same value. In the position of Figure 2.36 it is X's turn. If X plays the bottom-left square, X completes the left column and wins at once. If X plays the center, X creates two separate threats; O blocks one, X completes the other and wins two moves later. Both moves have the value +1.

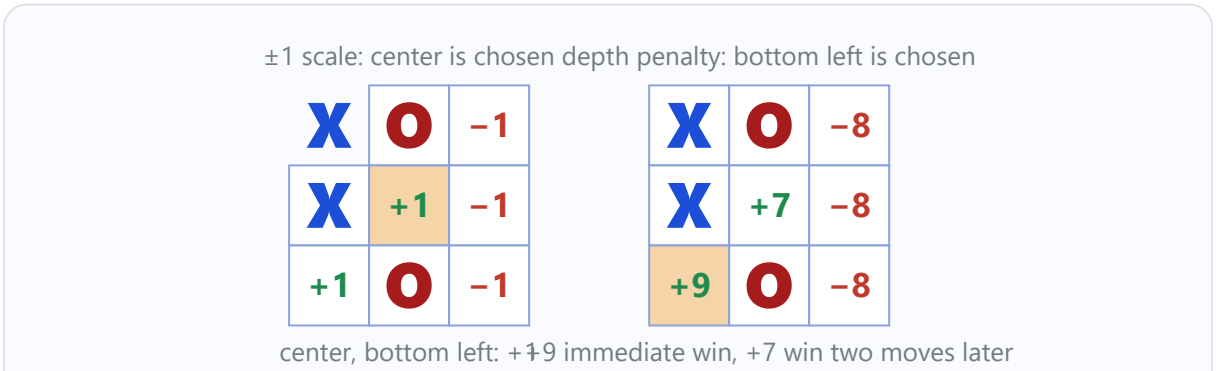


Figure 2.36: The same tic-tac-toe position on two scales. On the ±1 scale the center and bottom-left are equal (+1); with the depth penalty the bottom-left gets +9 (immediate win) and the center +7 (win two moves later). The other squares lose on both scales.

When values are equal, the order in which squares are tried decides the choice: in a tie the program keeps the first square it found. If the center is tried first, the program passes up the ready win and prolongs the game. The result does not change, X still wins; but the game is needlessly prolonged, and in a program that is not flawless this leaves room for error. On the losing side the situation is worse: when all moves get −1, the program may choose the move that loses fastest and does not even leave the opponent a chance to err.

The solution is to build into the utility how long the game takes. Let d be the number of moves made since the root. The utility of the leaves is defined as follows:

```
X wins      +(10 - d)
O wins      -(10 - d)
draw          0
```

This is called the **depth penalty**. The earlier a win comes, the larger the value: a win after one move from the root is +9, a win after three moves is +7, and since +9 > +7 the fast win is chosen. The later a loss comes, the larger the value: a loss after two moves is −8, a loss after four moves is −6, and since −6 > −8 the program delays the loss. Since a tic-tac-toe game lasts at most 9 moves, $10 - d$ is always positive; a win or a loss never crosses to the other side of a draw (0). The penalty changes not the type of the outcome but only the order among outcomes of the same type.

Example 2.38

In the position of Figure 2.36, X is on the top left and middle left, O is on the top middle and bottom middle; it is X's turn. The values of the five empty squares are computed on both scales:

1. **Bottom left.** The left column is completed and X wins one move from the root: +1 on the ±1 scale, $10 - 1 = +9$ with the penalty.
2. **Center.** X creates two threats: the left column (bottom left) and the diagonal (top left – center – bottom right). O blocks one, and X completes the other on the third move: +1, or $10 - 3 = +7$.
3. **Top right, middle right, bottom right.** O has a threat too: the top middle and bottom middle are O's, and the center is empty. If X plays on one of these three squares, O plays the center and completes the middle column. O wins two moves from the root: −1 on the ±1 scale, $-(10 - 2) = -8$ with the penalty.

On the ±1 scale the bottom left and the center are equal; with the depth penalty the bottom left clearly comes first. The program wins at once.

2.9.6 Justifying a move and the value of the empty board

Minimax chooses by numbers alone; it does not say why the choice was made. The justification of the chosen move can be found by asking, in order, what the move accomplishes:

1. Does the move complete a line of three? If so, the move **wins**.
2. Does it block a line the opponent would complete in one move? If so, the move **blocks**.
3. Does it create two separate winning threats at once? If so, the move **forks**: the opponent cannot block both threats.

4. If none of these, the sign of the value gives the reason: if positive it forces a win, if zero it **holds the draw**, if negative it delays the loss.

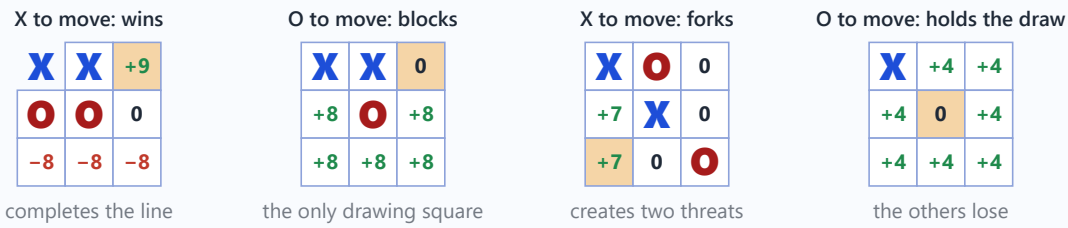


Figure 2.37: Four positions and the justification of the chosen move (values from X's point of view, with the depth penalty). From left to right: X completes the line and wins (+9); O blocks the threat with the only square that holds the draw (0, the other squares +8); X forks (+7); against a corner opening O answers with the center and holds the draw (0, the other seven squares +4).

Example 2.39

The two middle positions of Figure 2.37 are examined in detail.

1. **Block.** X is on the top left and top middle, O is at the center; it is O's turn. X has a threat on the top row: if the top right stays empty, X fills it and wins. O playing the top right closes the threat and keeps the value of the game at 0. Each of the other five squares gets +8: if O leaves the top right, X wins on the next move, that is, two moves from the root (10 − 2 = 8). Since O is MIN, it chooses the smallest of the values, 0.
2. **Fork.** X is on the top left and center, O is on the top middle and bottom right; it is X's turn. The middle left and bottom left get +7, the other squares 0. If X plays the bottom left, two threats arise at once: in the left column (top left – middle left – bottom left) and on the diagonal (bottom left – center – top right). O can block only one; X completes the other on the third move: 10 − 3 = +7.

A good tic-tac-toe player can also be written with rules such as "first win, then block, then fork". Minimax knows none of these rules; it finds all of them itself by searching the tree. The rules are specific to the game and do not carry over to another game; the same minimax program, however, works on Connect Four too.

A full minimax search from the empty board gives all nine of X's first moves the value 0 (Figure 2.38). No opening forces a win: if both sides play without error, the game ends in a draw. Since all values are equal, it is not the values but the order in which squares are tried that determines which square X opens with. If the order center, corners, edges is used, the program opens in the center.

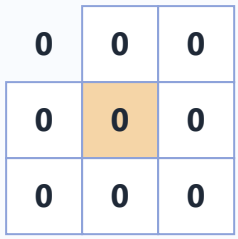


Figure 2.38: The empty tic-tac-toe board: the minimax value of all nine first moves is 0.

Holding the draw is not equally hard for the two sides. If X opens with a corner, only one of O's eight replies, the center, holds the draw; each of the other seven replies has the value +4 (Figure 2.37, the rightmost position). The value +4 shows that X wins six moves from the root, that is, with the fourth X stone: *10 − 6 = 4*. An O that answers a corner opening with an edge or another corner inevitably falls into a fork as long as X plays correctly.

2.9.7 Alpha-beta pruning: α, β, and the cutoff condition

In the code trace it was seen that C's second leaf was read needlessly. The same situation arises in every tree. In the tree of Figure 2.39, A (MAX) has been guaranteed 3 from its first child B: B = min(3, 12, 8) = 3. C's first leaf is 2. Since C is a MIN node, its value will be at most 2: MIN can find no reason to choose anything larger than 2. And A, holding 3, never chooses C, which gives at most 2. Whatever C's remaining leaves, 4 and 6, are, the value of the root does not change. These two leaves are never read; they are **pruned**.

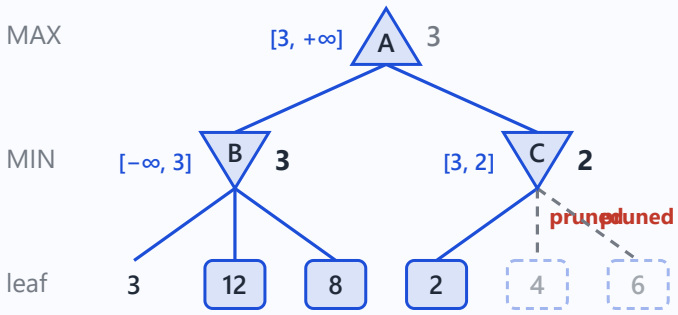


Figure 2.39: Pruning at a MIN node. A is guaranteed 3 from B. Once C's first leaf 2 is seen, C's value is at most 2; A will not choose C, and C's remaining leaves (4 and 6) are pruned. The interval [α, β] is written to the left of each node.

In a small tree skipping two leaves may seem unimportant. But in real games a "leaf" is often itself a deep subtree: in chess a single pruned branch saves searching millions of positions. **Alpha-beta pruning** makes this observation systematic with two numbers.

Definition 2.22

During the search, on the path from the root to a node:

- **α (alpha)** is the best value MAX has *already guaranteed* on this path: MAX's lower bound. It starts at −∞ and grows only at MAX nodes: $\alpha = max(\alpha, value)$.
- **β (beta)** is the best (smallest) value MIN has already guaranteed on the same path: MIN's upper bound. It starts at +∞ and shrinks only at MIN nodes: $\beta = min(\beta, value)$.

Each node takes its interval [α, β] from its parent and passes it unchanged to its children; only the node's value returns to the parent. When $\alpha \geq \beta$ at a node, the node's remaining children are not searched: the **cutoff condition**.

The meaning of the interval is this: if a node's value falls outside the interval [α, β], it does not affect the result at the root. If the value is smaller than α, a MAX node above will not choose this branch, since it has already guaranteed α from another branch. If the value is larger than β, a MIN node above will not choose this branch, since it has already guaranteed β from another branch. When the interval empties, that is, when $\alpha \geq \beta$, there is no region left in which the node's value could be "interesting".

In Figure 2.39 the interval arriving at C is [3, +∞]: α = 3 is the value A has guaranteed from B. When C reads its first leaf, β = min(+∞, 2) = 2. Now α = 3 ≥ β = 2 and C's remaining children are pruned. Pruning happens at a MAX node with the same condition. In the tree of Figure 2.40 the root is MIN and has been guaranteed 6 from B; at the root β = 6 and the interval [−∞, 6] arrives at C. C is a MAX node and its first leaf is 7: α = 7 ≥ β = 6. C gives at least 7; since the root, being MIN, has already guaranteed 6, it will not choose C. C's second leaf 4 is pruned.

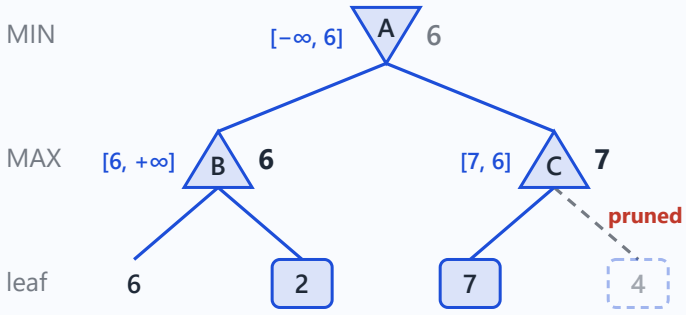


Figure 2.40: Pruning at a MAX node. The root is MIN and is guaranteed 6 from B (β = 6). C is a MAX node; once its first leaf 7 is seen, α = 7 ≥ β = 6. C gives at least 7, MIN will not choose C, and leaf 4 is pruned.

	At a MIN node	At a MAX node
Updated	β = min(β, value)	α = max(α, value)
Cutoff	when β ≤ α	when α ≥ β

	At a MIN node	At a MAX node
Meaning	this node gives at most β; the MAX above already guarantees α	this node gives at least α; the MIN above already guarantees β
Returned	an upper bound (true value ≤ β)	a lower bound (true value ≥ α)

Table 2.28: Pruning at a MIN and a MAX node. The two cases have the same condition: α ≥ β.

Some books call the cutoff at a MIN node an *α cutoff* and the one at a MAX node a *β cutoff*: the naming follows which bound caused the cutoff. Both cutoffs are tested with the same inequality; a single line `if a >= b` therefore suffices in the code.

2.9.8 Why pruning does not change the result

Alpha-beta finds the same root value and a best move as minimax (the same move if the move order is the same); it only reads fewer leaves. The truth of this claim is shown by knowing what the numbers returned by interior nodes mean. The number returned by a pruned node may not be the true minimax value; but compared with the true value, which side it stands on is known.

Theorem 2.3

Let the true minimax value of a node be v, and let r be the value returned by alpha-beta when called with the interval $[\alpha, \beta]$ ($\alpha < \beta$). Then:

- if $\alpha < v < \beta$, then $r = v$ (exact value);
- if $v \le \alpha$, then $r \le \alpha$ (it is known only to be "bad enough");
- if $v \ge \beta$, then $r \ge \beta$ (it is known only to be "good enough").

In particular, when the root is called with the interval $(-\infty, +\infty)$, the returned value is the minimax value of the root.

Proof. The proof is by induction on the height of the node. At a leaf $r = v$ and all three cases hold. Consider a MAX node (a MIN node is symmetric). The node tries its children in order and after each child updates α with the largest value returned so far.

As long as there is no cutoff, each child is called with the current interval $[\alpha', \beta]$ ($\alpha' \ge \alpha$). By the induction hypothesis, if a child's true value is greater than α' and less than β, the returned value is exact; if the true value does not exceed α', the returned value does not exceed α' either and does not change the maximum of the node. So every child that could determine the node's maximum is seen with its exact value. If all children finish without a cutoff, the value returned by the node is the true value itself when the true value is greater than α; if the true value does not exceed α, the returned value does not either.

If there is a cutoff, the value returned from a child has reached β. The true value of the MAX node cannot be smaller than the true value of this child, and by the induction hypothesis the true value of this child is not smaller than β. So $v \geq \beta$, and the returned value is not smaller than β either: the third case holds. Whatever the values of the unread children, this result does not change, because they can only increase the value of the MAX node. At the root $\alpha = -\infty$ and $\beta = +\infty$, so the true value is always inside the interval and the first case applies. □

In everyday language the theorem says this: whatever the true value of a pruned branch, it cannot change the result of the parent's max or min operation. For MAX, a value that cannot exceed α cannot take α's place; for MIN, a value that cannot go below β cannot take β's place. The number returned at interior nodes may be a bound, but the move chosen at the root is still the move minimax chooses.

2.9.9 Alpha-beta step by step: a twelve-leaf tree

The twelve-leaf tree solved by hand with minimax (Figure 2.33) is now searched with alpha-beta, trying the children from left to right. Each row is an event: descending into a node, reading a leaf, or a cutoff.

Step	Node	Event	Value	[α, β]
1	A (MAX)	the search starts	−∞	[−∞, +∞]
2	B (MIN)	the interval comes from A	+∞	[−∞, +∞]
3	E (MAX)	leaves 5 and 6 are read: max(5, 6)	6	[6, +∞]
4	B	E = 6: β = min(+∞, 6)	6	[−∞, 6]
5	F (MAX)	the interval comes from B; leaf 7: α = 7	7	[7, 6]
6	**F**	**α = 7 ≥ β = 6: leaf 4 is pruned**	**7**	**—**
7	B	F = 7: min(6, 7); B finishes	6	[−∞, 6]
8	A	B = 6: α = max(−∞, 6)	6	[6, +∞]
9	C (MIN)	the interval comes from A	+∞	[6, +∞]
10	G (MAX)	leaves 3 and 2 are read: max(3, 2); α stays 6	3	[6, +∞]
11	C	G = 3: β = min(+∞, 3)	3	[6, 3]
12	**C**	**β = 3 ≤ α = 6: all of H (8 and 9) is pruned**	**3**	**—**
13	A	C = 3: max(6, 3)	6	[6, +∞]
14	D (MIN)	the interval comes from A	+∞	[6, +∞]
15	I (MAX)	leaves 4 and 1 are read: max(4, 1); α stays 6	4	[6, +∞]
16	D	I = 4: β = min(+∞, 4)	4	[6, 4]
17	**D**	**β = 4 ≤ α = 6: all of J (2 and 7) is pruned**	**4**	**—**
18	A	D = 4: max(6, 4); the search ends	6	[6, +∞]

Table 2.29: Alpha-beta on the twelve-leaf tree. The [α, β] column is the interval at that node after the event.

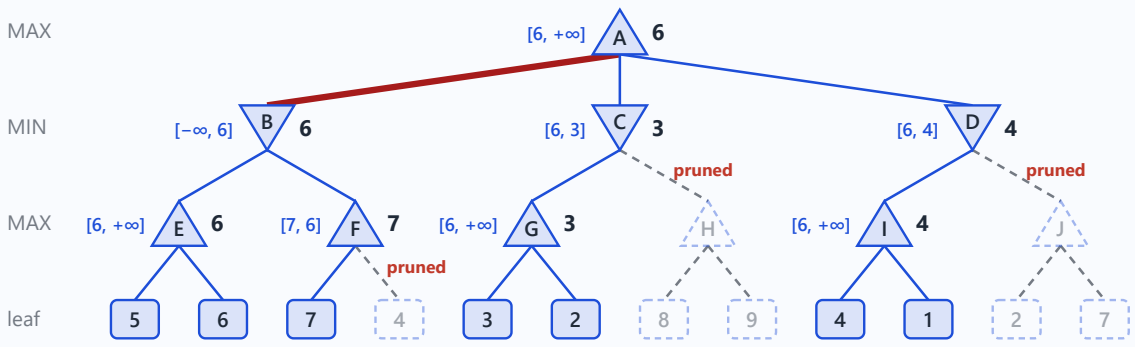


Figure 2.41: The end of the alpha-beta search. To the left of each node is the final [α, β] interval and to the right its value. Dashed lines and faded nodes are the pruned branches: F's second leaf, and all of H and J. Seven of the twelve leaves have been read.

The result is the same as minimax: the value of the root is 6, the chosen move A → B. The leaves read are 5, 6, 7, 3, 2, 4, 1; seven of the twelve leaves are read and five are pruned. Three points in the trace deserve attention:

- **The 7 returned by F is a bound.** F stopped without reading 4; the number it returned carries the information "F is at least 7". In this tree F's true value is also 7 (max(7, 4) = 7), but alpha-beta does not know this and does not need to: what matters for B is that F is larger than 6.
- **α does not rise at G and I.** The interval [6, +∞] arrives at G; its leaves are 3 and 2, and since *max(6, 3) = 6*, α does not change. The 3 returned by G is a value below α: the "bad enough" case of Theorem 2.3. C's β drops to 3 and the interval empties.
- **Pruning can remove a whole subtree.** What is pruned at C, H, is not a single leaf but a subtree with two leaves. In deep trees the pruned subtrees are much larger; the gain comes from here.

2.9.10 Code trace: the alpha-beta function

Adding two parameters and three lines to the minimax code gives alpha-beta. The same six-leaf tree is used.

```
from math import inf
T = {"A": ["B", "C", "D"], "B": [3, 5],
     "C": [2, 9], "D": [6, 4]}
def alphabeta(n, a, b, maxi):
  if isinstance(n, int):        # leaf
    return n
  best = -inf if maxi else inf
  for c in T[n]:
    v = alphabeta(c, a, b, not maxi)
    if maxi:
      best = max(best, v); a = max(a, best)
    else:
      best = min(best, v); b = min(b, best)
    if a >= b:
      break                     # cutoff
  return best
print(alphabeta("A", -inf, inf, True))
```

The differences from minimax are:

- The parameters `a` and `b` are α and β. The root is called with `-inf` and `inf`. The recursive call passes the node's current `a` and `b` to the child.
- A MAX node updates `a` after updating `best`, and a MIN node updates `b`. The updates are on local variables; in Python, changing `a` inside a function does not change the caller's `a`. In this way the interval is passed only downward, automatically.
- The line `if a >= b: break` is the pruning: the loop is exited without trying the remaining children.

Step	Call (incoming a, b)	Event	best	a, b (after)
1	A (−∞, +∞)	the root is opened	−∞	−∞, +∞
2	B (−∞, +∞)	leaf 3: min; a < b, continue	3	−∞, 3
3	B	leaf 5: min(3, 5); B finishes, returns 3	3	−∞, 3
4	A	B = 3: max; a = 3	3	3, +∞
5	C (3, +∞)	leaf 2: min; b = 2	2	3, 2
6	**C**	**a = 3 ≥ b = 2: `break`, leaf 9 is not read; returns 2**	**2**	**—**
7	A	C = 2: max(3, 2); a stays 3	3	3, +∞
8	D (3, +∞)	leaf 6: min; b = 6; a < b, continue	6	3, 6
9	D	leaf 4: min(6, 4); b = 4; a < b; D finishes, returns 4	4	3, 4
10	A	D = 4: max(3, 4); a = 4; A finishes	4	4, +∞

Table 2.30: Trace of `alphabeta("A", -inf, inf, True)`.

The program again prints `4`. Five of the six leaves are read; the 9 that minimax read needlessly has been pruned. There is no pruning at D: the interval arriving at D is [3, +∞] and D's value 4 lies inside the interval. D's value is computed exactly and raises the root's value from 3 to 4. This is the first case of Theorem 2.3: if the value is inside the interval, the exact value is returned.

2.9.11 Alpha-beta by hand: an eight-leaf tree

When applying alpha-beta by hand, three things are tracked for each interior node: the interval arriving from above, the node's current value, and whether the interval has emptied. The tree in Figure 2.42 shows this trace from start to finish.

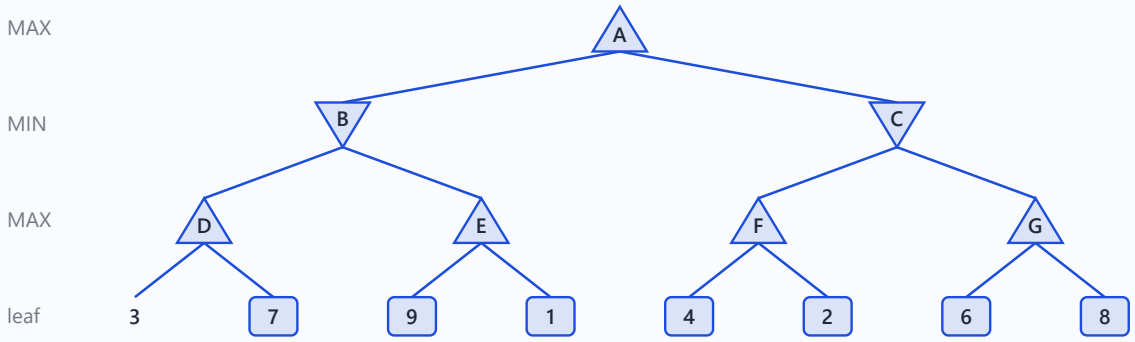


Figure 2.42: An eight-leaf tree: root A (MAX), B and C (MIN), D, E, F, G (MAX). The leaves are D = [3, 7], E = [9, 1], F = [4, 2], G = [6, 8].

Example 2.40

First the value of the root is found with minimax: D = max(3, 7) = 7, E = max(9, 1) = 9, B = min(7, 9) = 7; F = max(4, 2) = 4, G = max(6, 8) = 8, C = min(4, 8) = 4; A = max(7, 4) = 7. The value of the root is 7 and MAX chooses B. Then alpha-beta is applied, trying the children from left to right:

1. [−∞, +∞] arrives at B, and at D. D (MAX) reads 3 and 7: D = 7. B (MIN) gets β = min(+∞, 7) = 7.
2. [−∞, 7] arrives at E. The first leaf is 9: α = 9 ≥ β = 7. E gives at least 9; since B has already guaranteed 7, it will not choose E. Leaf 1 is pruned. B = min(7, 9) = 7.
3. At the root α = 7. [7, +∞] arrives at C, and at F. F (MAX) reads 4 and 2: F = 4 (α stays 7, because max(7, 4) = 7).
4. At C, β = min(+∞, 4) = 4 ≤ α = 7. C gives at most 4; since the root has already guaranteed 7, it will not choose C. All of G (6 and 8) is pruned.
5. Root = max(7, 4) = 7: MAX chooses B.

Five of the eight leaves (3, 7, 9, 4, 2) are read and three (1, 6, 8) are pruned. The result is the same as minimax. The 8 among G's leaves is not a number that could change C's true value: C's value is at most 4, the value of F, and this is below the 7 the root already holds.

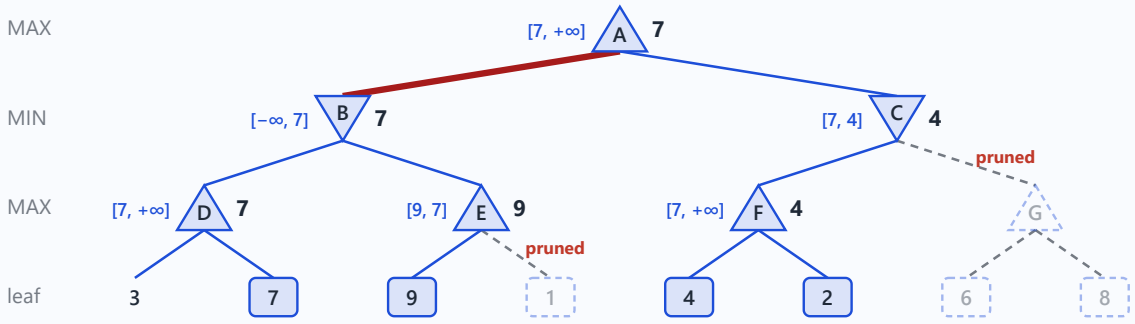


Figure 2.43: The end of the hand alpha-beta solution. E's second leaf (1) and all of G are pruned; the root chooses B with value 7.

Table 2.31 compares minimax and alpha-beta on three small trees and two real games. In every row the root value and the chosen move are the same; what changes is only the number of leaves read or nodes visited. The gain grows as the tree grows: a few leaves in small trees, about 70 times in tic-tac-toe, 189 times at depth 7 in Connect Four.

Tree	Minimax	Alpha-beta	Root value	Pruned, or order
Twelve-leaf tree	12 leaves	7 leaves	6 and 6	F's 4; all of H and J
Tree in the code trace	6 leaves	5 leaves	4 and 4	C's 9
Eight-leaf tree	8 leaves	5 leaves	7 and 7	E's 1; all of G
Tic-tac-toe, empty board	549,946 nodes	7,865 nodes	0 and 0	in the order center, corners, edges
Connect Four, start, depth 7	960,793 nodes	5,086 nodes	+8 and +8	in order from the middle outward

Table 2.31: Minimax and alpha-beta on the same tree.

2.9.12 Move ordering

How much alpha-beta prunes depends on the order in which children are tried; the value of the root does not depend on the order. In the eight-leaf tree, had C been tried before B, the root would have had to compute C's value exactly, because at that point it holds no lower bound. When the best child is tried first, α and β narrow early and the later siblings are quickly eliminated.

This effect was measured by Knuth and Moore (1975). In a uniform tree of depth d in which every interior node has b children:

- In the **best order** (at a MAX node the child with the largest value first, at a MIN node the smallest), the number of leaves read is $b^{\lceil d/2 \rceil} + b^{\lfloor d/2 \rfloor} - 1 \approx b^{d/2}$.

- In the **worst order** there is no pruning at all and all b^d leaves are read.

The structure of the formula comes from an observation. In the best order, all the children of the nodes on the path from the root to the best child must be opened, because the exact values of these nodes are needed. For the others, a single "refuting" reply below a sibling suffices to show that the sibling is not better. For this reason in half of the levels of the tree only one child is opened and in the other half all children are, and the number of leaves read grows with the half power of b. In the same time one can descend to about **twice the depth**; in other words, the effective branching factor falls from b to $\sqrt{b}$.

d	35^d (no pruning)	Best order
2	1,225	69
4	1,500,625	2,449
6	1,838,265,625	85,749
8	$\approx 2.25 \cdot 10^{12}$	3,001,249

Table 2.32: Number of leaves read for b = 35 (the average number of moves in chess).

The numbers in the table are computed directly from the formula. For example, for $d = 4$, $35^2 + 35^2 - 1 = 1{,}225 + 1{,}225 - 1 = 2{,}449$; for $d = 2$, $35 + 35 - 1 = 69$. At depth 8, while a search without pruning reads more than 2 trillion leaves, alpha-beta in the best order makes do with 3 million leaves. The work of best-order alpha-beta at depth 8 is about twice the work of an unpruned search at depth 4.

The effect is visible in a small tree as well. In a tree of 27 leaves with depth 3 and branching factor 3, $3^2 + 3^1 - 1 = 11$ leaves are read in the best order; in the worst order there is no pruning at all and all 27 leaves are read. The order changes only the order in which children are tried; the tree is the same tree and the value of the root is the same in every order.

Knowing the best order means already knowing the best move; in practice the order is guessed. There are three sources of good guesses:

- **Game knowledge:** in Connect Four the middle column, in tic-tac-toe the center and corners, in chess moves that capture a piece first.
- **The previous search:** the move that an earlier, shallower search found best is tried first.
- **Experience at sibling nodes:** a move that caused a cutoff in another branch is tried first in this branch too.

In tic-tac-toe the effect of order can be counted directly. Table 2.33 gives the number of nodes visited by full searches from the empty board (with the depth-penalty scale).

Method	Nodes visited	Root value
Minimax (order does not matter)	549,946	0
Alpha-beta, squares in order 0–8 (left to right, top to bottom)	20,866	0

Method	Nodes visited	Root value
Alpha-beta, center first, then corners, then edges	7,865	0
Minimax with a transposition table (each position once)	5,478	0

Table 2.33: Full search from the empty tic-tac-toe board: number of nodes visited and root value.

Good ordering reduces the number of nodes by a factor of about 2.7 (20,866 / 7,865); compared with minimax the gain is about 70-fold (549,946 / 7,865). The value scale also affects pruning: the same two alpha-beta searches with the ±1 scale visit 18,297 and 7,275 nodes. The coarser the scale, the more values are equal; equal values satisfy the condition $\alpha \geq \beta$ more often, so pruning increases. The depth penalty, in contrast, demands finer ordering and searches more nodes (about 2,600 more in the 0–8 order, about 600 more with good ordering); in return it distinguishes a fast win from a slow one and a late loss from an early one.

2.9.13 Connect Four: why full search is impossible

Connect Four is played on an upright board with 7 columns and 6 rows. A player chooses a column and the disc falls to the lowest empty place in that column. The first player to line up four of their discs horizontally, vertically, or diagonally wins; if the board fills up, the game is a draw. The branching factor is at most 7 and the length of the game at most 42 plies. By John Tromp's count the number of legal positions is 4,531,985,219,092, that is, about $4.5 \cdot 10^{12}$.

The game was solved in 1988, independently, by James D. Allen and Victor Allis: if the first player starts in the middle column, they can force a win. This solution was found not by brute force but with game-specific rules and very strong search. It is not a job that can be done in a browser in a second. Even a minimax that looks only 7 plies ahead from the starting position visits 960,793 nodes.

When a full search is impossible, the search is **stopped** at a certain depth. The positions where it stops are not game endings; their utilities are unknown. In their place an **evaluation function** estimates how good the position is.

2.9.14 Depth-limited search and the evaluation function

Definition 2.23

An **evaluation function** *E(s)* gives a position that is not a game ending a score from MAX's point of view: large if the position looks good for MAX, small if it looks bad. **Depth-limited search** cuts minimax off so that it looks at most *d* plies ahead:

```
H(s, d) =
  utility(s)                  if s is terminal
  E(s)                        if d = 0
  max  H(result(s, a), d - 1) if MAX to move
   a
  min  H(result(s, a), d - 1) if MIN to move
   a
```

The first two lines of the definition together are called the **cutoff test**: if the game is over or the depth limit has been reached, descending stops. *E(s)* is not an exact value but an estimate; minimax uses these estimates as if they were leaves and propagates them up to the root. Alpha-beta works the same way: the *E* values at the limit are read like leaves.

The values of the evaluation function and the values of real game endings should lie in separate ranges. The value of a game ending must be larger than every value *E* can take (for a win) or smaller (for a loss); otherwise the program may prefer a position that "looks good" to a real win. The utility used here is: if Red (MAX) wins at ply *k* from the root, *+(1000 − k)*; if Blue (MIN) wins, *−(1000 − k)*; if the board fills, 0. The evaluation function takes at most a few hundred. The depth penalty of tic-tac-toe is present here too: as *k* gets smaller the value of a win grows, and the later a loss comes the larger the value of the loss.

A simple and effective evaluation function for Connect Four is based on the **windows of four** on the board. A window is a place made up of four adjacent squares in which a four could be completed. On a 7 × 6 board there are 69 windows: 24 horizontal (4 in each row, 24 over six rows), 21 vertical (3 in each column, 21 over seven columns), and 24 diagonal (12 in each of the two directions). Each window gets a score according to the discs in it.

Discs in the window	p(w)
3 red + 1 empty	+5
2 red + 2 empty	+2
3 blue + 1 empty	−5
2 blue + 2 empty	−2

Discs in the window	p(w)
both colors, a single disc, or empty	0

Table 2.34: The score p(w) of a window of four.

The evaluation function adds to the sum of the window scores a bonus for the middle column:

```
E(s) = Σ p(w) + 3 · (R_mid − B_mid)
```

Here R_{mid} and B_{mid} are the numbers of red and blue discs in the middle column. The reason for rewarding the middle column is geometric: a square in the middle column belongs to between 7 and 13 windows, while a square in an edge column belongs to only 3 to 5. A disc in the middle can be part of more fours. A window containing both colors scores 0, because in that window nobody can make a four any more.

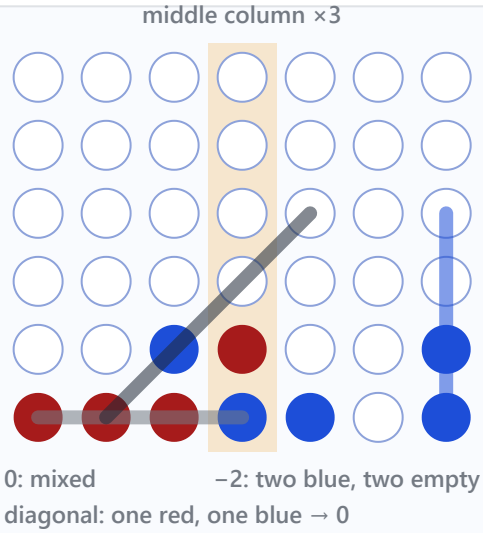


Figure 2.44: Example windows on a Connect Four board, with the middle column highlighted. A window with two blue discs and two empty squares scores −2; windows containing both colors score 0.

Example 2.41

Red has played in the middle column (column 4), Blue in the column to its left (column 3), and Red in the column to the right of the middle (column 5); all three discs are in the bottom row. *E(s)* is computed as follows:

1. **Horizontal windows of the bottom row.** The windows of columns 1–4, 2–5, and 3–6 contain the blue disc together with at least one red disc: 0. The window of columns 4–7 has two red discs (4 and 5) and two empty squares (6 and 7): +2.

2. **The other windows.** All vertical and diagonal windows contain at most one of the discs; a window with a single disc scores 0. Windows containing only the blue disc also have a single disc: 0.
3. **Middle column.** There is one red disc in the middle column: $3 \cdot (1 - 0) = +3$.

The total is $E(s) = 2 + 3 = +5$: the position looks slightly good for Red. The same function also explains the value of a depth-1 search from the empty board: when Red plays the middle column, a single disc scores in no window, only the middle-column bonus remains, and $E = +3$. A disc in an edge column gets 0. At depth 1 the root value is +3 and the program chooses the middle column.

The code of depth-limited alpha-beta adds to the code for the abstract tree only the cutoff test and move generation:

```
from math import inf
def alphabeta(s, d, a, b, maxi):
  if s.is_over():              # game over
    return s.utility()         # ±(1000 - k) or 0
  if d == 0:
    return E(s)                # evaluation
  best = -inf if maxi else inf
  for m in order(s.moves()):
    v = alphabeta(s.play(m), d-1, a, b, not maxi)
    if maxi:
      best = max(best, v); a = max(a, best)
    else:
      best = min(best, v); b = min(b, best)
    if a >= b:
      break
  return best

def best_move(s, d):       # root: MAX
  a, top = -inf, None
  for m in order(s.moves()):
    v = alphabeta(s.play(m), d-1, a, inf, False)
    if v > a:
      a, top = v, m
  return top
```

The game-specific parts of the code are five: `is_over`, `utility`, `moves`, `play`, and `E`. The rest is independent of the game; the same skeleton can be used for tic-tac-toe, Connect Four, or chess. `order` puts the moves in the order in which they will be tried; in Connect Four this order is from the middle outward: 4, 3, 5, 2, 6, 1, 7. The function `best_move` keeps at the root not only the value but the

move. The α at the root is carried across the root's children: the first child is searched with the full interval, the later ones with the narrowing window [α, +∞], and it is quickly shown that they are no better than the first.

2.9.15 The horizon effect

Depth-limited search cannot see events just beyond the limit. This is called the **horizon effect**. In the tree of Figure 2.45, by the evaluations at the depth limit the position under B looks best with E = +6. Yet one move after that position there is an unavoidable loss (around −1000). Since the search stops at the limit, it does not see this loss and chooses B.

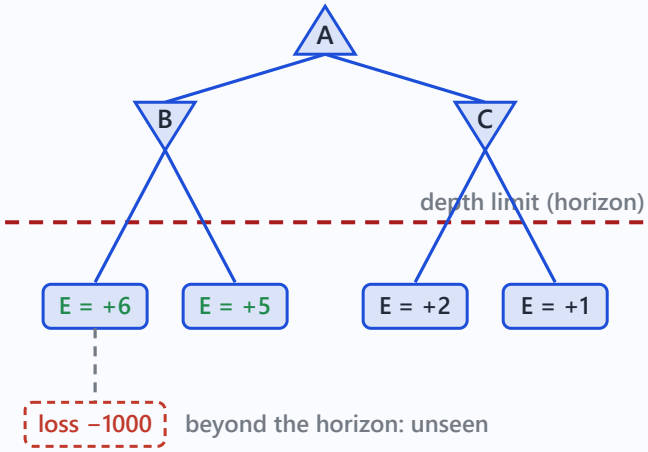


Figure 2.45: The horizon effect. By the evaluation made at the depth limit (dashed line), the position under B is best with E = +6; the loss one move beyond the limit is invisible to the search.

There is a more insidious form of the horizon effect: the program makes meaningless moves that push an unavoidable loss beyond the horizon. In chess, a program that sees that a piece will be lost delays the loss by a few plies by giving pointless checks or sacrificing a pawn. The loss does not disappear; it only becomes invisible because it falls outside the search depth; and, on top of that, the sacrificed pawn is lost too.

There is no complete cure for the horizon effect; there are methods that reduce its effect:

- **Increasing the depth:** the horizon moves away but does not disappear.
- **Quiescence search:** if the position at the limit is "active", for example a capture or a threat is in progress, the search continues with only such moves and the position is evaluated when it has calmed down.
- **Threat extension:** while one side has a threat to win in one move, the search descends one more level.

A program playing Connect Four with depth 1 or 2 cannot see double threats and is easily beaten: the double threat the opponent will set up two moves later lies outside the program's horizon. As the depth is increased, the program starts to see these traps. That the root value goes up and down with depth as +3, 0, +6, 0, +6, 0, +8 is also a consequence of limited vision: at odd depths Red makes the last move at the horizon and at even depths Blue does, and the side making the last move appears in the evaluation with an extra disc.

2.9.16 Move ordering, iterative deepening, and the transposition table

The effect of move ordering in Connect Four is measured with searches at different depths from the starting position (Table 2.35).

Depth	Minimax	Alpha-beta (left to right)	Alpha-beta (middle outward)	Root value
1	8	8	8	+3
2	57	33	21	0
3	400	200	76	+6
4	2,801	966	175	0
5	19,608	4,340	627	+6
6	137,257	16,409	1,285	0
7	960,793	67,549	5,086	+8

Table 2.35: Number of nodes visited in a search from the Connect Four starting position.

Minimax grows roughly as 7^d: at depth 1, 8 nodes (the root and seven children), at depth 2, $1 + 7 + 49 = 57$, at depth 7, 960,793. At depth 7 alpha-beta visits about 14 times fewer nodes in left-to-right order (960,793 / 67,549) and 189 times fewer in middle-outward order (960,793 / 5,086). The root value is the same in all three methods in every row: the order changes only the cost. Since the middle column takes part in the most fours, good moves are mostly in the middle; when they are tried first, α and β narrow early.

Iterative deepening solves the problem of choosing the search depth in advance. First a search with depth 1 is run, then 2, 3, …; when time runs out, the move of the last completed search is played. At first sight repeating shallow searches seems wasteful, but the repeated work is small: in an exponentially growing tree the last level is larger than the sum of the earlier ones. For $b = 7$, the total work of all depths is only about $7/6$ times the work of the last depth alone:

```
7 + 7² + … + 7^d ≈ 7^d · (1 + 1/7 + 1/49 + …) = 7^d · 7/6
```

The second gain of iterative deepening is in ordering: the best move of the previous search is tried first in the next. Move ordering thus improves by itself, and the cost of the shallow searches is more than recovered by the pruning gained in the deep search.

A **transposition table** prevents the same position from being searched over and over. The same position can be reached by different move orders (a *transposition*). A position is searched once and the result is written into a hash table; when the position is met again the result is read from the table. In tic-tac-toe a full search thus makes do with 5,478 positions instead of 549,946 nodes (Table 2.33). An entry in the table contains: the value, the depth to which the position was searched, whether the value is exact or a lower or upper bound, and the best move. The bound information is needed because of alpha-beta: the value returned from a pruned search may not be exact (Theorem 2.3). Chess programs turn a position into a single number, a hash key, with a method proposed by Zobrist (1970); a move updates this key with a few XOR operations.

The skeleton of strong game-playing programs consists of these parts: iterative deepening, alpha-beta, good move ordering, a transposition table, and a good evaluation function. Real systems use this skeleton in different proportions:

- **Deep Blue (1997).** IBM's chess computer beat the world champion Garry Kasparov 3½–2½ in a six-game match. It ran parallel alpha-beta on special chess chips and evaluated about 200 million positions per second.
- **Chess engines.** Programs such as Stockfish still use a search derived from alpha-beta: iterative deepening, a transposition table, and strong pruning methods. Since 2020 the evaluation function has been a neural network (NNUE).
- **Go and Monte Carlo tree search.** In Go $b \approx 250$ and writing a good $E(s)$ is very hard. Monte Carlo tree search (Coulom, 2006; UCT: Kocsis and Szepesvári, 2006) evaluates positions with randomly played games. AlphaGo combined this search with deep neural networks and beat Lee Sedol 4–1 in March 2016.

The common idea of these systems is the same as minimax: looking ahead while assuming the opponent's best reply. What changes is how much of the tree is searched and how the leaves are evaluated.

2.9.17 Common mistakes in search algorithms

Common mistake

- **Passing α and β in the wrong direction.** The interval descends unchanged from parent to child; only the value returns to the parent. An α or β updated in a child is not carried to the parent. MAX updates only α, MIN only β. In the twelve-leaf tree α = 7 at F, but only F's value returns to B; B's interval stays [−∞, 6].
- **Counting a pruned branch.** The values of pruned leaves are not read; feeding them into a min or max operation in a hand solution is wrong. The number returned by a node that was cut off is also not an exact value but a bound.
- **Thinking the root value depends on the order.** Move ordering changes only the number of leaves read. If a different order gives a different root value, there is a bug in the code.

- **Viewpoint and sign errors.** *E(s)* must always be written from the point of view of the same player (here Red, that is, MAX). If the program is the second player, it chooses the smallest. If negamax is written, the sign is flipped at every level and the evaluation is made from the point of view of the player to move; if the two formulations are mixed up, the program plays for its opponent.
- **Forgetting the depth penalty.** If all wins get the same value, the program postpones a ready win; and when losing it may choose the fastest loss. The utility must also include how many moves later the win comes.
- **Keeping the evaluation on the same scale as game endings.** If *E(s)* can reach the value of a real win, the program may prefer a good-looking position to a sure win. The values of game endings should lie outside every value the evaluation function can take.
- **Taking a depth-limited value as exact.** *H(s, d)* is an estimate from a look of *d* plies. In Connect Four the +8 at depth 7 does not mean "Red wins"; the real outcome of the game is known only by a full search or a proof.

2.10 The algorithmic view and mechanism design

Most of the concepts so far belong to classical game theory. But a thread has also emerged throughout the chapter: minimax, backward induction, and Nim's XOR rule are algorithms; in the 2/3 game the elimination is a loop. This place where game theory meets computer science gives rise to the questions of algorithmic game theory.

2.10.1 Why is algorithmic game theory "algorithmic"?

Three observations make the algorithmic view necessary.

Size. Even the description of a game grows exponentially. If each of n players has 2 strategies, the number of strategy profiles is 2^n. Since each player's payoff must be written for each profile, the matrix contains $n \cdot 2^n$ numbers. In a game with 30 players the number of profiles is $2^{30} \approx 1.07 \cdot 10^9$. A brute-force method that checks all profiles one by one does not scale.

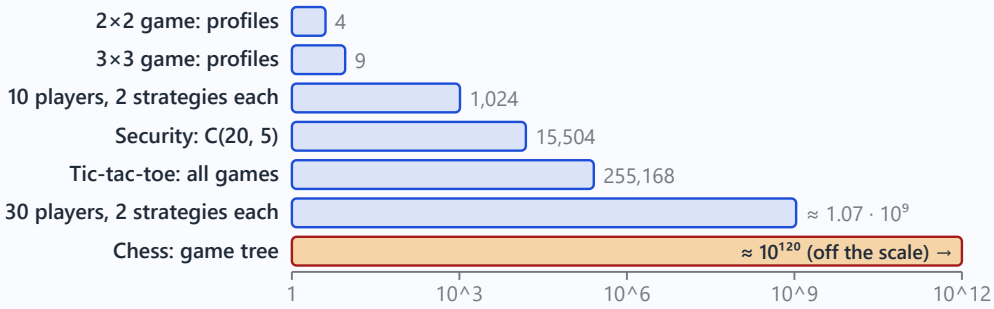


Figure 2.46: The sizes of games on a logarithmic scale. A 2×2 game has 4 profiles and a 3×3 game 9; a game with 10 players and two strategies each has 1,024, protecting 5 of 20 targets C(20, 5) = 15,504, tic-tac-toe has 255,168 games, and a game with 30 players and two strategies each has about $1.07 \cdot 10^9$ profiles. The chess tree ($\approx 10^{120}$) does not fit the scale.

Each bar in Figure 2.46 is many times larger than the one before. In the security example, each choice of 5 targets to protect out of 20 is a strategy; the number of choices is *C(20, 5) = 20! / (5! · 15!) = 15,504*. These numbers show that most games can be solved only with algorithms that exploit their structure.

Complexity. By Nash's theorem every finite game has an equilibrium in mixed strategies. But knowing that an equilibrium exists does not mean finding it. Even for two-player games no fast method of finding an equilibrium is known: the problem (exact equilibrium for two players, approximate equilibrium for many players) belongs to the class of hard problems called PPAD-complete (Daskalakis, Goldberg, and Papadimitriou; for two players, Chen and Deng).

Structure and design. Special structure brings efficiency. Zero-sum games are solved quickly by linear programming; in Nim, XOR labels exponentially many positions with a single computation. Rules are algorithms too: the rule that determines the winner and the payment in an auction is a program, and it must be designed correctly.

2.10.2 Four questions

Algorithmic game theory has four main questions. The concepts of this chapter, that is, the classification of games, the equilibria of classic games, and backward induction in sequential games, are the common language of all four.

1. **Computation: can an equilibrium be found?** Zero-sum games are solved quickly by linear programming. For general games no fast method for finding a Nash equilibrium is known; the problem is PPAD-complete. Combinatorial games can also be hard: solving Hex on large boards is PSPACE-complete (Reisch, 1981).
2. **Efficiency: what does selfishness cost?** The **Price of Anarchy** is the ratio of the cost of the worst equilibrium to the cost of the best outcome. In the Prisoner's Dilemma, if years in prison are counted as cost, the equilibrium is 4 years in total and (Silent, Silent) is 2 years; the ratio is 4 / 2 = 2.

3. **Design: which rules?** Mechanism design looks for rules that make players report their private information truthfully, that is, rules that are **incentive compatible**: auctions, matching, voting.
4. **Dynamics: how do players reach equilibrium?** In repeated and online games, information arrives over time; players adapt their strategies by learning from the past. The guesses falling toward 0 over repeated rounds of the 2/3 game are an example of this.

Example 2.42

In the price competition game (Table 2.4) the profits are payoffs, not costs. In this case the efficiency loss is measured by the ratio of the total profit of the best outcome to the total profit at equilibrium:

1. Total profit at equilibrium: (100, 100) = 5 + 5 = 10.
2. The outcome that maximizes total profit: (120, 120) = 6 + 6 = 12. (In the other cells 8 + 1 = 9.)
3. Ratio: 12 / 10 = 1.2.

Selfish pricing costs the firms about 17% of their total profit (from 12 to 10, 2/12). The ratio in the Prisoner's Dilemma (2) is larger; the loss depends on the numbers in the game.

2.10.3 Mechanism design: game theory in reverse

Game theory predicts the outcome of a given game: the rules are fixed, and the question is what will happen. **Mechanism design** turns the question around: the desired outcome is fixed, and the question is what rules and incentives will produce that outcome. For this reason mechanism design is also called "reverse game theory".

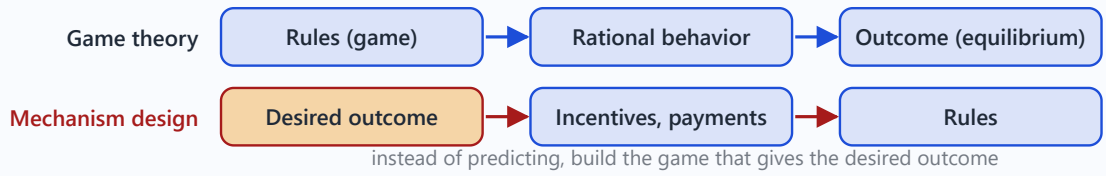


Figure 2.47: Game theory and mechanism design. Top row: from the rules to rational behavior, and from there to the outcome (the equilibrium). Bottom row: from the desired outcome to incentives and payments, and from there to the rules.

The difficulty of design is this: the agents' private information, for example the value they place on an item, their location, or their preferences, is hidden from the designer. Agents may lie when reporting this information. A good mechanism makes lying unprofitable. As Vickrey showed in 1961, in a second-price auction bidding truthfully is a dominant strategy: the winner pays not their own bid but the second-highest bid; inflating or shading the bid therefore adds nothing to the payoff.

Mechanisms fall into two groups:

- **With money:** the designer can collect or make payments; the payments encourage truth-telling. Examples: auctions, ad placement.
- **Without money:** incentives are provided only by the rule; usually the goal is compromised a little. Examples: elections, house swaps, facility location.

2.10.4 Real-world applications

Mechanism design and algorithmic game theory work inside many real systems:

- **Search ads:** ad slots are allocated in every search by a generalized second price (GSP) auction. Unlike the single-slot second-price auction, in multi-slot GSP truthful bidding is not dominant.
- **Spectrum auctions:** governments sell frequency licenses in designed auctions; in the United States this practice has continued since 1994.
- **School placement:** in the student-proposing Gale–Shapley algorithm it is best for a student to report their true preferences; the result is a stable matching.
- **Room and shift swaps:** everyone brings what they hold; through swap cycles nobody gets something worse than before and no money is used.
- **Traffic and routing:** each driver chooses their own fastest path; the cost of selfish choices to total delay is measured by the Price of Anarchy.
- **Platforms and incentives:** surge pricing draws drivers to busy hours; Bitcoin miners validate transactions for a reward.

The common idea of these applications is this: the rules are designed so that the system works well even if everyone acts in their own interest. In the Prisoner's Dilemma individual interest led to a bad shared outcome; in a well-designed mechanism individual interest leads to the desired outcome.

2.11 Solved problems

Five problems bring together the main methods of the chapter; the solutions are given with intermediate steps.

Example 2.43

Problem. In the price competition game (Table 2.4), what are the dominant strategies and the equilibrium? Is the equilibrium Pareto optimal?

Solution. For Firm 1, 100 TL gives 5 > 1 against Firm 2's 100 and 8 > 6 against its 120: 100 is dominant. Since the game is symmetric, 100 is dominant for Firm 2 as well. The equilibrium is (100, 100) = (5, 5). Because (120, 120) = (6, 6) gives both firms more, it Pareto dominates the equilibrium; the equilibrium is not Pareto optimal.

Example 2.44

Problem. In the 2/3 game (0–90), if 20% of the class thinks at level 0 and 80% at level 1, which level wins?

Solution. The expected average is $0.2 \cdot 45 + 0.8 \cdot 30 = 9 + 24 = 33$ and the target is $2/3 \cdot 33 = 22$. The distances: level 0 (45) 23, level 1 (30) 8, level 2 (20) 2, level 3 (13.3) 8.7. Level 2 is closest. The Nash answer, 0, is 22 away.

Example 2.45

Problem. Two players choose A or B. (A, A) = (4, 1), (B, B) = (1, 4), the other cells (0, 0). What are the pure equilibria? Which classic game does this game resemble?

Solution. In column A the row player gets 4 with A and 0 with B → A. In column B, 0 with A and 1 with B → B. In row A the column player gets 1 with A and 0 with B → A. In row B, 0 with A and 4 with B → B. The cells with two marks are (A, A) and (B, B). One favors the row player (4 > 1), the other the column player (4 > 1). Meeting is a shared interest and where to meet is a conflict: the structure of Battle of the Sexes.

Example 2.46

Problem. A MAX root has three MIN children: B = [7, 2, 9], C = [4, 6], D = [1, 8]. What is the value of the game and what does MAX choose?

Solution. The MIN values: B = min(7, 2, 9) = 2, C = min(4, 6) = 4, D = min(1, 8) = 1. The root MAX = max(2, 4, 1) = 4; MAX chooses C. The 9 at B and the 8 at D are misleading: MIN does not allow them.

Example 2.47

Problem. In the market entry game, why is (Stay out, Fight) not subgame perfect?

Solution. In the subgame that starts after E has entered, Fight gives I −1 and Share gives 1. Fight is not a best response in this subgame; hence the profile does not form a Nash equilibrium in that subgame. "If you enter, I will fight" is a non-credible threat. The profile is a Nash equilibrium of the whole game, but not of every subgame.

2.12 Summary

- A **game** consists of players, strategies, payoffs, information, and rules. A **rational** player maximizes expected utility given their beliefs; rational does not mean selfish. **Common knowledge** is the chain of "everyone knows" continuing forever.
- A **strategy** is a complete plan saying what to do at every decision node; m actions at each of k nodes give m^k strategies. A **mixed strategy** is a probability distribution over pure strategies.
- **Ordinal utility** carries only the order, **cardinal utility** also the size of differences. Lottery and mixed-strategy computations require cardinal utility; the transformation $a \cdot u + b$ ($a > 0$) preserves decisions.
- The **normal form** $G = (N, (S_i), (u_i))$ writes simultaneous games as a matrix; the **extensive form** writes sequential games as a tree. An information set shows a simultaneous game in a tree.
- Games are classified along six dimensions: timing, agreements, total payoff, symmetry, information, and horizon. **Perfect information** concerns past moves, **complete information** concerns payoff functions.
- A **Nash equilibrium** is a profile from which nobody wants to deviate unilaterally. In the Prisoner's Dilemma Betray is dominant and the only equilibrium, (−2, −2), is not **Pareto optimal**: stability is not efficiency.
- The **six classic 2×2 games** range from conflict to coordination: Matching Pennies has no pure equilibrium; Chicken, Battle of the Sexes, Stag Hunt, and Pure Coordination have two pure equilibria; in Free Money one of them is weak.
- In the **2/3 game** iterated elimination 90 → 60 → 40 → … → 0 gives the unique equilibrium (everyone 0); real players think 1–2 steps and the best guess depends on beliefs (**level-k**).
- **Minimax** computes values from the leaves to the root; tic-tac-toe is a draw, the example tree has value 3. In **Nim** a position is losing if its nim-sum is 0 (Bouton); in Don't Say 21 the key numbers are multiples of 4.
- **Subgame perfect equilibrium** is a Nash equilibrium in every subgame and is found by backward induction; it rules out non-credible threats (market entry, investment, ultimatum). In the Centipede game the SPE is (1, 0).
- A **game tree** reaches a size of roughly b^d. The **minimax** value rises from the leaves to the root: $O(b^d)$ time, $O(b \cdot d)$ memory. In Nim, minimax, the P/N labels, and the XOR rule find a winning move (**negamax**); in tic-tac-toe the empty board has value 0 and the **depth penalty** $\pm(10 - d)$ makes the program choose the fast win and the late loss.

- **Alpha-beta pruning** passes the interval [α, β] from top to bottom and skips the remaining children when $\alpha \geq \beta$; the root value and the chosen move do not change (7 leaves instead of 12 in the twelve-leaf example). With the best **move ordering** about $b^{d/2}$ leaves are read (Knuth–Moore, 1975); in tic-tac-toe 7,865 nodes suffice instead of 549,946.
- In games where full search is impossible, the search stops at a depth limit and an **evaluation function** estimates the position. Being unable to see beyond the limit is the **horizon effect**; **iterative deepening** and a **transposition table** speed up the search. In Connect Four at depth 7, middle-outward alpha-beta visits 189 times fewer nodes than minimax.
- The **grim trigger strategy** deters betrayal in infinite repetition with a permanent punishment; with the Sam–John payoffs cooperation is an equilibrium when $\delta \geq 1/2$. With a finite, known horizon cooperation unravels.
- Algorithmic game theory asks four questions: computation, efficiency (the Price of Anarchy), design (mechanism design), and dynamics.

Concept	Short definition	Example
Normal / extensive form	A matrix (simultaneous) or a game tree showing order and information	RPS, price competition; chess
Classification	Timing, agreements, total payoff, symmetry, information, horizon	Chess: sequential, zero-sum, perfect information
Dominant strategy, Nash equilibrium	Better whatever the other side does; no profitable unilateral deviation	Betray; (Betray, Betray)
Pareto optimality	Cannot be improved without making someone worse off	The equilibrium (−2, −2) is not
Classic 2×2 games	Conflict or coordination; pure equilibria found with best responses	Matching Pennies: none; Stag Hunt: two
Iterated elimination, level-k	Deleting the dominated again and again; best guess given beliefs	2/3 game: 90 → 60 → 40 → … → 0
Minimax, Nim	Best choice from leaves to root; P/N positions	Tree value 3; tic-tac-toe draw; Nim XOR
Alpha-beta, depth-limited search	Skipping branches that cannot affect the result with [α, β]; evaluation function at the depth limit	7 of 12 leaves; tic-tac-toe: 549,946 → 7,865 nodes
Subgame perfect equilibrium	Nash in every subgame; found by backward induction	Market entry; Centipede (1, 0)
Grim trigger strategy	Deterring betrayal in infinite repetition with a permanent punishment	Cooperation among patient players (large δ)

Table 2.36: Concept map of the chapter.

2.13 Key terms

game theory player rationality common knowledge strategy strategy profile pure strategy mixed strategy payoff utility function expected utility ordinal utility cardinal utility lottery information set perfect information complete information incomplete information Bayesian game

normal form payoff matrix extensive form game tree leaf zero-sum game constant-sum game maximin minimax symmetric game coalition discount factor best response Nash equilibrium dominant strategy Prisoner's Dilemma Pareto optimal Pareto dominance Price of Anarchy commitment focal point level-k reasoning iterated elimination of dominated strategies combinatorial game Zermelo's theorem strategy-stealing argument nim-sum subgame subgame perfect equilibrium backward induction non-credible threat centipede game ultimatum game grim trigger branching factor ply negamax alpha-beta pruning move ordering evaluation function depth-limited search horizon effect quiescence search iterative deepening transposition table Monte Carlo tree search mechanism design incentive compatible

2.14 Test Yourself

1. Firm 1 chooses one of four prices (80, 90, 100, 120 TL). Firm 2 sees this price and says 90, 100, or 120 TL. How many pure strategies does each firm have?

   Answer

   Firm 1 has a single decision node and 4 actions: 4 strategies. Firm 2 has 4 decision nodes (one for each of Firm 1's prices) and at each chooses among 3 actions: $3^4 = 81$ strategies. The number of actions (3) or nodes × actions (12) is not the number of strategies.

2. In the cells (u_1, u_2): (Top, Left) = (2, 2), (Top, Right) = (0, 3), (Bottom, Left) = (3, 0), (Bottom, Right) = (1, 1). What are the dominant strategies, the Nash equilibrium, and the Pareto optimal outcomes?

   Answer

   For the row player Bottom is dominant: 3 > 2 against Left, 1 > 0 against Right. For the column player Right is dominant: 3 > 2 against Top, 1 > 0 against Bottom. The only Nash equilibrium is (Bottom, Right) = (1, 1). (Top, Left) = (2, 2) Pareto dominates the equilibrium. The Pareto optimal outcomes are (2, 2), (0, 3), and (3, 0). The game is a Prisoner's Dilemma.

3. In the 2/3 game (0–90), if 60% of the class thinks at level 0 and 40% at level 1, what is the target? Is level 1 or level 2 closer to the target?

   Answer

   The expected average is $0.6 \cdot 45 + 0.4 \cdot 30 = 27 + 12 = 39$ and the target is $2/3 \cdot 39 = 26$. Level 1 (30) is 4 away from the target and level 2 (20) is 6 away. In this population level 1 wins: in a class where the others think little, thinking too much loses.

4. A MAX root has three MIN children: B = [5, 9], C = [6, 1, 8], D = [7, 4]. What is the minimax value of the game and what does MAX choose? Why can't the largest leaf in the tree be chosen?

   Answer

   The MIN values: B = min(5, 9) = 5, C = min(6, 1, 8) = 1, D = min(7, 4) = 4. The root MAX = max(5, 1, 4) = 5; MAX chooses B. The largest leaf 9 is at B, but at B the choice is MIN's and MIN chooses 5.

5. The Nim position is [1, 3, 6]. Is the player to move in a winning position? What is the winning move?

   Answer

   $1 \oplus 3 = 001 \oplus 011 = 010 = 2$; $2 \oplus 6 = 010 \oplus 110 = 100 = 4$. The nim-sum is $4 \neq 0$: it is a winning position. The heaps are tried: $1 \oplus 4 = 5$ (larger than 1), $3 \oplus 4 = 7$ (larger than 3), $6 \oplus 4 = 2$ (smaller than 6). The only winning move is to take 4 tokens from the third heap, leaving [1, 3, 2]: $1 \oplus 3 \oplus 2 = 0$.

6. In "Don't Say 25" players take turns saying 1 to 4 consecutive numbers; whoever says 25 loses. Which states (last number said) are losing, and who wins?

   Answer

   The base state is 24: the player to move must say 25 (L). From 23, 22, 21, 20 one can reach 24 (W). All of the states 20–23 reachable from 19 are W: 19 is L. The pattern continues in steps of 5: the L states are 24, 19, 14, 9, 4. Since 4 can be reached from the start 0, the first player wins: they say "1, 2, 3, 4", and then, if the opponent says x numbers, they say $5 - x$ numbers.

7. In the market entry game, suppose the Incumbent gains a reputation from fighting: let the leaf Enter–Fight be (−1, 3) instead of (−1, −1); the other leaves are the same (Stay out (0, 2), Enter–Share (1, 1)). What is the subgame perfect equilibrium? Is the threat "if you enter, I will fight" now credible?

   Answer

   At I's node Fight gives 3 and Share gives 1; I fights and the node's value is (−1, 3). At the root, Enter gives E −1 and Stay out gives 0; E stays out. The SPE is (Stay out, Fight) with outcome (0, 2). The threat is now credible, because when E enters, fighting really is in I's interest.

8. In a repeated game the payoffs each round are: if both cooperate, 3 each; if one deviates while the other cooperates, the deviator gets 6 and the other 0; if both deviate, 1 each. Both players play grim trigger. For deviating to be unprofitable, what must the probability δ that the game lasts one

more round be at least?

Answer

Complying: $3 / (1 - \delta)$. Deviating: 6 this round, then 1 every round: $6 + \delta / (1 - \delta)$. The condition is $3 / (1 - \delta) \geq 6 + \delta / (1 - \delta)$. Multiplying both sides by $(1 - \delta)$: $3 \geq 6(1 - \delta) + \delta = 6 - 5\delta$, that is, $\delta \geq 3/5$. As the temptation to deviate grows (a larger number than 6), the threshold grows too.

9. A MAX root has three MIN children: B = [4, 8, 6], C = [2, 7], D = [5, 3, 9]. If the children are tried from left to right, which leaves does alpha-beta read and which does it prune? What is the value of the root?

Answer

All three of B's leaves are read: B = min(4, 8, 6) = 4 and at the root $\alpha = 4$. [4, +∞] arrives at C; with the first leaf 2, $\beta = 2 \leq \alpha = 4$ and 7 is pruned. [4, +∞] also arrives at D; with 5, $\beta = 5$ and the interval is still nonempty; with 3, $\beta = 3 \leq 4$ and 9 is pruned. The leaves read are 4, 8, 6, 2, 5, 3 (six of eight). The value of the root is max(4, 2, 3) = 4 and MAX chooses B. The result of minimax is the same: C = 2, D = 3.

10. In a uniform tree of depth 4 in which every interior node has 4 children, how many leaves does minimax read? How many does alpha-beta read with the best move ordering?

Answer

Minimax reads all the leaves: $4^4 = 256$. In the best order the Knuth–Moore formula gives $4^2 + 4^2 - 1 = 31$. In the worst order there is no pruning and again 256 leaves are read; the root value is the same in all three cases.

11. In tic-tac-toe the depth penalty $\pm(10 - d)$ is used. Of X's three moves, one forces a win 3 plies from the root, one forces a win 7 plies later, and one leads to a draw. What are the values of the moves and which does X choose? What changes on the ±1 scale?

Answer

The values are $10 - 3 = +7$, $10 - 7 = +3$, and 0; X chooses the largest, that is, the fast win (+7). On the ±1 scale both wins get +1 and the choice between them is decided only by the order in which the moves are tried; the program may needlessly prolong the win.

12. In Connect Four Red has put two discs in the middle column (first and second rows from the bottom); Blue's only disc is at the bottom of the leftmost column. With the window scores and the

middle-column bonus, what is $E(s)$?

Answer

The vertical window in the middle column covering rows 1–4 from the bottom has two red discs and two empty squares: +2. All the other windows contain at most one of the red discs, or only the blue disc: 0. The middle-column bonus is $3 \cdot (2 - 0) = 6$. The total is $E(s) = 2 + 6 = +8$.

13. In a game with branching factor 3, iterative deepening searches from depth 1 up to 8. About how many times the search at depth 8 alone is the total work? What do you see when comparing with b = 7?

Answer

The total work is $3 + 3^2 + \ldots + 3^8 \approx 3^8 \cdot (1 + 1/3 + 1/9 + \ldots) = 3^8 \cdot 3/2$: about 1.5 times. For b = 7 the ratio is 7/6 ≈ 1.17. In general the ratio is $b / (b - 1)$: the larger the branching factor, the more the last level exceeds the sum of the earlier ones, and the smaller the extra cost of iterative deepening.

2.15 Further reading

- M. J. Osborne, *An Introduction to Game Theory* (2004), Chapters 2–5: normal-form games, the Nash equilibrium, extensive-form games, and backward induction, explained with little mathematics and many examples.
- A. R. Karlin and Y. Peres, *Game Theory, Alive* (2017): combinatorial games, Nim, and the strategy-stealing argument, with detailed proofs at student level.
- C. F. Camerer, *Behavioral Game Theory* (2003): examines the behavior of real players in games such as the 2/3 game, Centipede, and Ultimatum with experimental results.
- S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach* (4th edition, 2020), Chapter 5: explains minimax, alpha-beta pruning, evaluation functions, and Monte Carlo tree search through game-playing programs.
- D. E. Knuth and R. W. Moore, "An analysis of alpha-beta pruning" (1975): the classic paper that proves the correctness of alpha-beta and the number of leaves read in the best ordering.
- N. Nisan, T. Roughgarden, É. Tardos, and V. Vazirani (eds.), *Algorithmic Game Theory* (2007), Chapter 1: introduces the basic concepts of game theory in terms of the questions of algorithmic game theory.

2.16 References

1. L. V. Allis, *A Knowledge-Based Approach of Connect-Four*, master's thesis, Vrije Universiteit Amsterdam, 1988.
2. C. L. Bouton, “Nim, a game with a complete mathematical theory”, *Annals of Mathematics* 3, 1901–1902.
3. C. F. Camerer, *Behavioral Game Theory*, Princeton University Press, 2003.
4. M. Campbell, A. J. Hoane, F. Hsu, “Deep Blue”, *Artificial Intelligence* 134, 57–83, 2002.
5. R. Coulom, “Efficient selectivity and backup operators in Monte-Carlo tree search”, *Computers and Games*, 2006.
6. D. Easley, J. Kleinberg, *Networks, Crowds, and Markets*, Cambridge University Press, 2010.
7. W. Güth, R. Schmittberger, B. Schwarze, “An experimental analysis of ultimatum bargaining”, *Journal of Economic Behavior & Organization* 3(4), 1982.
8. R. A. Hearn, E. D. Demaine, *Games, Puzzles, and Computation*, A K Peters, 2009.
9. A. R. Karlin, Y. Peres, *Game Theory, Alive*, American Mathematical Society, 2017.
10. D. E. Knuth, R. W. Moore, “An analysis of alpha-beta pruning”, *Artificial Intelligence* 6(4), 293–326, 1975.
11. L. Kocsis, C. Szepesvári, “Bandit based Monte-Carlo planning”, *ECML*, 2006.
12. H. W. Kuhn, “Extensive games and the problem of information”, *Contributions to the Theory of Games* II, Princeton University Press, 1953.
13. R. D. McKelvey, T. R. Palfrey, “An experimental study of the centipede game”, *Econometrica* 60(4), 1992.
14. R. Nagel, “Unraveling in guessing games: An experimental study”, *American Economic Review* 85(5), 1995.
15. J. F. Nash, “Equilibrium points in n-person games”, *PNAS* 36(1), 1950.
16. N. Nisan, T. Roughgarden, É. Tardos, V. Vazirani (eds.), *Algorithmic Game Theory*, Cambridge University Press, 2007 (Chapter 1).
17. M. J. Osborne, *An Introduction to Game Theory*, Oxford University Press, 2004 (Chapters 2–5).
18. S. Reisch, “Hex ist PSPACE-vollständig”, *Acta Informatica* 15, 1981.
19. R. W. Rosenthal, “Games of perfect information, predatory pricing and the chain-store paradox”, *Journal of Economic Theory* 25, 1981.
20. T. Roughgarden, *Twenty Lectures on Algorithmic Game Theory*, Cambridge University Press, 2016.
21. S. Russell, P. Norvig, *Artificial Intelligence: A Modern Approach*, 4th edition, Pearson, 2020 (Chapter 5).
22. J. Schaeffer et al., "Checkers is solved", *Science* 317(5844), 2007.
23. T. C. Schelling, *The Strategy of Conflict*, Harvard University Press, 1960.
24. R. Selten, "Spieltheoretische Behandlung eines Oligopolmodells mit Nachfrageträgheit", *Zeitschrift für die gesamte Staatswissenschaft* 121, 301–324 and 667–689, 1965.
25. C. E. Shannon, “Programming a computer for playing chess”, *Philosophical Magazine* 41(314), 1950.
26. Y. Shoham, K. Leyton-Brown, *Multiagent Systems*, Cambridge University Press, 2009 (Chapters 3 and 5).
27. D. Silver et al., "Mastering the game of Go with deep neural networks and tree search", *Nature* 529, 484–489, 2016.

28. B. Skyrms, *The Stag Hunt and the Evolution of Social Structure*, Cambridge University Press, 2004.

29. J. Tromp, John's Connect Four Playground (number of Connect Four positions).

30. J. Tromp, G. Farnebäck, “Combinatorics of Go”, *Computers and Games*, 2006.

31. J. Tromp, “The Number of Legal Go Positions”, *Computers and Games*, 2016.

32. J. von Neumann, “Zur Theorie der Gesellschaftsspiele”, *Mathematische Annalen* 100, 1928.

33. E. Zermelo, "Über eine Anwendung der Mengenlehre auf die Theorie des Schachspiels", *Proceedings of the Fifth International Congress of Mathematicians*, vol. 2, 501–504, Cambridge University Press, 1913.

34. A. L. Zobrist, “A new hashing method with application for game playing”, Tech. Rep. 88, University of Wisconsin, 1970.

PART I · CHAPTER 3

Nash Equilibrium

In a game, each player's payoff depends not only on their own choice but also on the choices of the others. "What should I do?" therefore cannot be separated from "What will they do?" The Nash equilibrium is the concept that breaks this circle: when everyone correctly anticipates the others' choices, no one wants to change their own. These notes start from the definition and move on to methods of computation, the existence theorem, the problem of choosing among multiple equilibria, and zero-sum games. The last topic is how hard it is for a computer to find an equilibrium.

LEARNING OUTCOMES

Definition
Define the Nash equilibrium formally and relate it to the notion of best response.

Finding pure equilibria
Mark best responses in a payoff matrix, apply iterated elimination of dominated strategies, and find the equilibrium of games with continuous strategies such as Cournot and Bertrand.

Mixed equilibria
Compute a mixed equilibrium by hand and in code with the indifference principle; find the minimax value in zero-sum games and security games.

Existence and complexity
Interpret Nash's existence theorem, the problem of multiple equilibria, and the PPAD-completeness of computing an equilibrium.

3.1 Foundations

Every concept in this chapter revolves around a single question: when several decision makers' decisions are interdependent, what counts as a "reasonable" outcome? A small game and an everyday situation make the question concrete; the formal definition is built on top of them.

3.1.1 A first game: the Prisoner's Dilemma

Two suspects, Sam and John, are interrogated in separate rooms. Without knowing what the other does, each chooses one of two options: stay *Silent* or *Betray* the other. If both stay silent, each serves one year for lack of evidence. If both betray, each serves two years. If one betrays and the other stays silent, the betrayer goes free and the silent one serves three years.

In game theory the outcomes of a game are written as **payoffs**, and each player wants to make their own payoff as large as possible. With prison time, however, less is better. To reconcile the two views, the payoff is taken to be the negative of the years in prison: 1 year is a payoff of −1, and 3 years is a payoff of −3. The rule "larger is better" then holds everywhere; −1 is larger than −3.

A game is usually written as a **payoff matrix**. The rows are the options of one player (here Sam), the columns those of the other (John). Each cell holds two numbers: first the row player's payoff, then the column player's. This order does not change anywhere in these notes.

Sam \ John	Silent	Betray
Silent	(−1, −1)	(−3, 0)
Betray	(0, −3)	(−2, −2)

Table 3.1: The Prisoner's Dilemma. Each cell shows the payoffs (Sam, John); payoff = −(years in prison).

Reading the table is the most basic skill of this chapter. For example, the upper-right cell (Silent, Betray) = (−3, 0) says: Sam stayed silent and John betrayed; Sam serves 3 years (payoff −3) and John goes free (payoff 0). Sam can choose only the row; John's column determines which cell the game lands in. This is exactly what separates a game from a single-person decision problem.

Put yourself in Sam's place. What John will do is unknown. If John stays silent, Sam's options are −1 (Silent) versus 0 (Betray); betraying is better. If John betrays, the options are −3 (Silent) versus −2 (Betray); again betraying is better. So whatever John does, betraying is better for Sam. The same reasoning applies to John, because the game is symmetric. The result is that both betray and each serves two years, although if both had stayed silent they would have served only one year each. This is where the "dilemma" in the name comes from: each player's individually sensible choice leads to a bad outcome for both.

When the game is repeated, the picture changes. In the version played against a computer, the computer can use the **Tit-for-Tat** strategy: it stays silent in the first round and in every later round copies the opponent's previous choice. Against such an opponent, what is the minimum number of years accumulated over ten rounds? The answer rests on the fact that betrayal is punished in the next round, and it is computed in Section 3.3.2.

3.1.2 John Nash and the origin of the concept

The concept takes its name from the American mathematician John Forbes Nash Jr. (1928–2015). In 1950 Nash wrote a short doctoral thesis at Princeton, *Non-Cooperative Games*. In the same year he published a two-page note in *PNAS* proving that every finite game has an equilibrium point (Nash, 1950). An expanded version of the thesis appeared the following year in the *Annals of Mathematics* (Nash, 1951).

Before Nash, the strongest result in game theory was John von Neumann's minimax theory of 1928 for two-player **zero-sum games** (von Neumann, 1928). In a zero-sum game one player's gain is the other's loss, so interests are exactly opposed. Most real-life interactions are not like that. In the Prisoner's Dilemma both players prefer the outcome (Silent, Silent) to (Betray, Betray); their interests are partly shared. Nash carried the idea of equilibrium over to general n-player games that are not zero-sum.

For this work Nash shared the 1994 Nobel Prize in Economics with John Harsanyi and Reinhard Selten. In 2015 he was awarded the Abel Prize for his contributions to partial differential equations. His life was the subject of the film *A Beautiful Mind* (2001).

3.1.3 Strategic behavior and applications

The basic assumption of game theory is that players try to **maximize** their own payoffs. Single-person decision theory works with the same assumption. What is new in game theory is that everyone's payoff also depends on the choices of the others. To make the best choice, a player must predict what the others will do, and the others in turn predict theirs. This is called **strategic behavior**.

Strategic dependence appears in many different areas. The table below summarizes examples that recur in this chapter and later ones, along with their structure.

Situation	What happens?	Structure
Price competition	Two firms cut prices; both lose profit.	Prisoner's Dilemma structure
Advertising / arms race	Everyone spends; relative positions do not change.	Dominant strategy: spend
Penalty kick	Kicker and goalkeeper must be unpredictable.	Mixed strategy, zero-sum
Choosing a lane in traffic	If everyone moves into the fast lane, it slows down.	Congestion game
Choosing a standard	Everyone gains by choosing the same technology.	Coordination, multiple equilibria
Resource sharing	Shared bandwidth, servers, roads.	Price of Anarchy

Table 3.2: Examples of strategic interaction and their structure in game theory.

In none of these examples can the "best choice" be stated independently of the opponent. If the goalkeeper always dives left, the kicker shoots right; if the kicker always shoots right, the goalkeeper dives right. When choosing a standard, the right technology is the one the others choose. In computer

networks, whether it pays for a device to send a packet at a given moment depends on whether the others are sending too. So what we look for is not each player's best choice in isolation but situations in which the choices are consistent with one another.

3.1.4 The situation nobody regrets

Before the formal definition, let us think about the concept through an everyday example. A cafeteria has two checkout counters, and students are spread across the queues. Everyone wants a shorter wait, and switching queues is allowed. Figure 3.1 shows two situations.

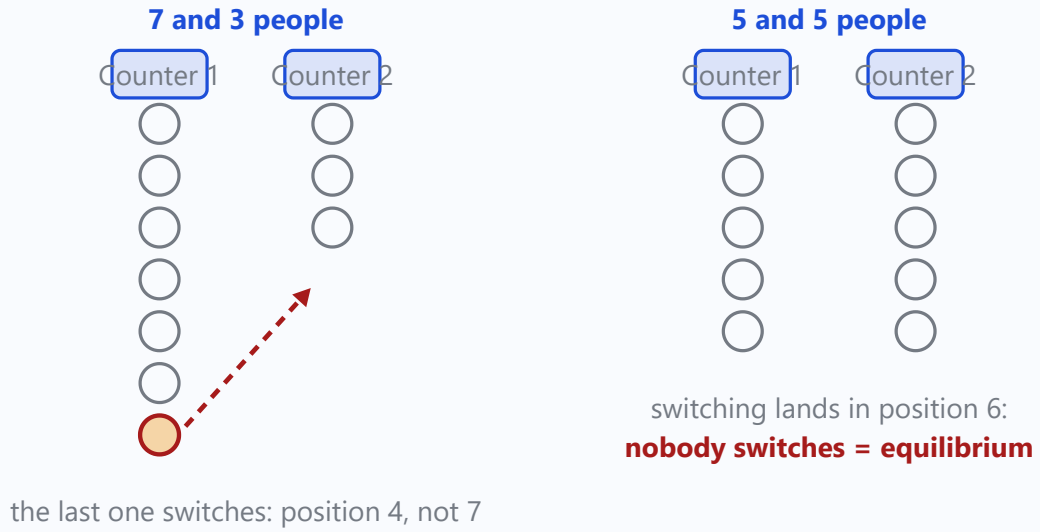


Figure 3.1: Two checkout counters in a cafeteria. Left: queues of 7 and 3 people; if the student at the end of the long queue switches, they move from position 7 to position 4, so switching pays. Right: queues of 5 and 5; a student who switches lands in position 6, so nobody switches.

If the queues have 7 and 3 people, the student at the end of the long queue moves to the other counter and goes from position 7 to position 4. Because switching pays, this situation is not lasting; someone is bound to move. If the queues have 5 and 5 people, a student who switches lands in position 6 of the other queue, that is, further back. Nobody has a reason to move; the situation sustains itself.

A test follows from this observation. Call it the **regret test**: "If I had seen what the others did, would I change my own choice?" If everyone's answer is "no", the situation is a Nash equilibrium. The test has two important features. First, each player considers changing only their *own* choice; the others' choices are held fixed. Second, the test is applied to each player separately; if even one person answers "yes", the situation is not an equilibrium.

An equilibrium is a self-sustaining situation: everyone correctly expects the others' behavior and does the best they can in response. But an equilibrium does not mean the "best outcome" or a "fair outcome". In the Prisoner's Dilemma, (Betray, Betray) passes the regret test, yet both players would

prefer (Silent, Silent). Equilibrium is a concept of stability, not a concept of evaluation.

3.1.5 The formal definition

To state the regret test precisely, the game itself must first be stated precisely. The notation below is used throughout the chapter.

Definition 3.1

A **normal-form game** (also called a strategic-form game) $G = (N, (S_i), (u_i))$ consists of three components:

- $N = \{1, \ldots, n\}$: the set of players.
- S_i: player i's **strategy set**, that is, the options available to that player.
- u_i: player i's **payoff function**. It assigns a number to every situation in which each player has made a choice.

A list in which each player has chosen a strategy is called a **strategy profile**: $s = (s_1, \ldots, s_n)$. The strategies of everyone other than player i are written s_{-i}; a profile can thus also be written $s = (s_i, s_{-i})$. $u_i(s)$ is player i's payoff when profile s is played.

In the Prisoner's Dilemma, $N = \{Sam, John\}$ and $S_i = \{Silent, Betray\}$ for both players. There are four strategy profiles, and each is a cell of the table. For example, for $s = (Silent, Betray)$ we have $u_{Sam}(s) = -3$ and $u_{John}(s) = 0$. From Sam's point of view, s_{-Sam} is simply John's strategy.

The notation s_{-i} may look odd at first. The idea is simple, though: when player i decides, they think of the others' choices as a single package. In a two-player game this package is a single strategy; in a three-player game it is a list of two strategies. The notation stays the same however many players there are.

Definition 3.2

A profile $s^* = (s_1^*, \ldots, s_n^*)$ is a **Nash equilibrium** (NE) if and only if, for every player i and every $s_i \in S_i$,

$$u_i(s_i^*, s_{-i}^*) \geq u_i(s_i, s_{-i}^*)$$

holds. In other words, while the others play s_{-i}^*, no player can increase their payoff by changing their strategy.

The left side of the inequality is the player's payoff at the equilibrium. The right side is the payoff they would receive if they switched their own strategy to s_i while the others stayed put. Such a change is called a **unilateral deviation**. The definition says that no unilateral deviation is profitable. Note the "≥" sign: a deviation may leave the payoff equal; it just cannot increase it.

The definition can be read in three ways:

- **Unilateral deviation:** With the others held fixed, no player can raise their payoff by changing their strategy.
- **Self-enforcing agreement:** If the players agreed in advance on a profile and everyone believes the others will keep to the agreement, each will keep to it too. No outside enforcement is needed to carry out the agreement.
- **Stability is not efficiency:** An equilibrium guarantees neither fairness nor efficiency. In particular, an equilibrium need not be **Pareto optimal**. An outcome is Pareto optimal if there is no other outcome that makes everyone at least as well off and at least one player strictly better off.

Example 3.1

The four cells of the Prisoner's Dilemma are checked one by one against the definition. For each cell two questions are asked: does the row player (Sam) gain by changing the row, and does the column player (John) gain by changing the column?

1. (Silent, Silent) = (−1, −1). If Sam switches to Betray, the cell becomes (Betray, Silent) and Sam gets 0. Since 0 > −1, the deviation is profitable. The cell is not an equilibrium; there is no need to look at John.
2. (Silent, Betray) = (−3, 0). If Sam switches to Betray he gets −2; −2 > −3. Not an equilibrium.
3. (Betray, Silent) = (0, −3). If John switches to Betray he gets −2; −2 > −3. Not an equilibrium.
4. (Betray, Betray) = (−2, −2). If Sam switches to Silent he gets −3; −3 < −2, so the deviation is unprofitable. If John switches to Silent he also gets −3; again unprofitable. For neither player does any deviation increase the payoff. The cell is a Nash equilibrium.

Conclusion: the game's only pure Nash equilibrium is (Betray, Betray), with payoffs (−2, −2), that is, two years in prison each. (Silent, Silent), although better for both, is not an equilibrium. This is why the equilibrium is not Pareto optimal.

Checking four cells one by one is easy in a small game but becomes tiring as the game grows. The best-response marking in Section 3.2 carries out the same check systematically.

3.1.6 Misconceptions about the definition

Because the definition of the Nash equilibrium is short, it is easy to misread. The most common misconceptions are listed below. In each case, the error can be seen from a single word in the definition.

Common mistake

- **"An equilibrium is the best outcome."** No. In the Prisoner's Dilemma the equilibrium (B, B) is worse for both players than (S, S). Equilibrium speaks of stability, not of efficiency.
- **"If both would gain by changing together, it is not an equilibrium."** The definition looks only at unilateral deviations. A joint deviation by two players (a coalition) does not destroy a Nash equilibrium. Moving together from (B, B) to (S, S) benefits both, but that lies outside the definition.
- **"It is enough if one player is content."** The test is applied to each player separately; if even a single player wants to deviate, the cell is not an equilibrium.
- **"The row player also chooses the column."** Each player changes only their own strategy. The row player moves up and down within the same column; the column player moves left and right within the same row.
- **"Every game has exactly one pure equilibrium."** Matching Pennies has no pure equilibrium; Stag Hunt has two pure equilibria and one mixed equilibrium. These are computed in Section 3.4 and Section 3.5.
- **Penalty or payoff?** For costs such as years in prison, the smaller number is better. To convert to a payoff, a minus sign is added: 2 years → −2. Choosing "the larger one" without the sign gives exactly the opposite answer.

3.2 Best response

The definition of the Nash equilibrium asks, for every profile, "is a deviation profitable?" To answer this systematically, one first finds, for each player, the best answer to each possible choice of the others. This answer is called the best response. Finding an equilibrium then comes down to finding where the best responses coincide.

3.2.1 Best response and mutual best response

The question behind the best response is simple: "If I knew what my opponent would do, what would I do?" The opponent's choice is fixed, the player looks only at their own payoffs, and picks the option that gives the largest one.

To write this down we use **argmax** notation. *max* gives the largest *value* in a list; *argmax* gives the *option* (or options) that attain this largest value. For example, if four options have payoffs A: 3, B: 7, C: 2, D: 7, then *max = 7*, but *argmax = {B, D}*. When the largest value occurs at more than one option, argmax is a set; this detail matters in games with ties.

Definition 3.3

Player i's **best response** to the others' choice s_{-i} is the set of strategies that maximize their payoff:

$$BR_i(s_{-i}) = \operatorname{argmax}_{s_i \in S_i} u_i(s_i, s_{-i})$$

A best response is not a constant; it is a rule that depends on the opponent's choice. When the opponent changes, the best response may change too. It helps to think of BR_i as "a table that tells me what to do against each choice of the opponent". This rule is also called the **best-response correspondence**. The word "correspondence" stresses that one input may be assigned several outputs (a set, in case of ties).

The inequality in the definition of the Nash equilibrium says that s_i^* gives the largest payoff against s_{-i}^*. That means s_i^* is a best response. The theorem below makes this observation precise.

Theorem 3.1 (Equilibrium and best response)

A profile s^* is a Nash equilibrium if and only if $s_i^* \in BR_i(s_{-i}^*)$ for every player i. In short: an equilibrium is a mutual best response.

Proof. We show the two directions separately. ($\Rightarrow$) Let s^* be a Nash equilibrium. By definition, $u_i(s_i^*, s_{-i}^*) \geq u_i(s_i, s_{-i}^*)$ for every s_i. This means the payoff from s_i^* is the largest payoff attainable against s_{-i}^*. A strategy that gives the largest payoff lies in the argmax set; hence $s_i^* \in BR_i(s_{-i}^*)$.

($\Leftarrow$) Suppose $s_i^* \in BR_i(s_{-i}^*)$ for every i. A strategy in the argmax set gives the largest payoff, so no other s_i can give a larger one. This is exactly the inequality in the definition. Since it holds for every player, s^* is a Nash equilibrium. □

The theorem gives a practical method for finding equilibria. In a two-player matrix game, three steps suffice:

1. **Row player:** In each column, mark the row player's largest payoff (the first number in the cell, u_1). Because the column is held fixed, this is the row player's best response to that choice of the column player.
2. **Column player:** In each row, mark the column player's largest payoff (the second number, u_2).
3. **Intersection:** The cells in which both payoffs are marked are the pure-strategy Nash equilibria.

The cost of the method is small. In an $m \times n$ matrix, each cell is compared with one column maximum and one row maximum. If the column maxima and row maxima are computed once, the total work is $O(mn)$, proportional to the number of cells in the matrix. When the matrix is already given, no faster method is possible, since every cell must be looked at least once.

3.2.2 The marking method: the Prisoner's Dilemma

Let us try the method first on the familiar game. The payoffs are the negatives of the years in prison (−1 = 1 year), and each player wants to make their own number larger. Label the cells a, b, c, d: a = (Silent, Silent), b = (Silent, Betray), c = (Betray, Silent), d = (Betray, Betray).

Example 3.2

1. **If John stays Silent**, Sam looks only at the first numbers in this column: Silent −1, Betray 0. Since 0 > −1, Sam's best response is Betray; the 0 in cell c is marked.
2. **If John Betrays:** Silent −3, Betray −2. Since −2 > −3, the best response is again Betray; the first −2 in cell d is marked.
3. In both columns the mark is in the same row. Whatever John does, Betray is better for Sam. Such a strategy is called a **dominant strategy** (Section 3.3.1).
4. **If Sam stays Silent**, John looks at the second numbers: Silent −1, Betray 0; the best response is Betray, and the 0 in cell b is marked.
5. **If Sam Betrays:** Silent −3, Betray −2; Betray for John as well. The second −2 in cell d is marked. Because the game is symmetric, this result is no surprise.
6. The only cell where both marks meet is d = (Betray, Betray). Nobody wants to deviate alone: a Nash equilibrium with payoffs (−2, −2).

Yet a = (Silent, Silent) would have given each player only one year. The equilibrium is not Pareto optimal; that is the essence of the dilemma.

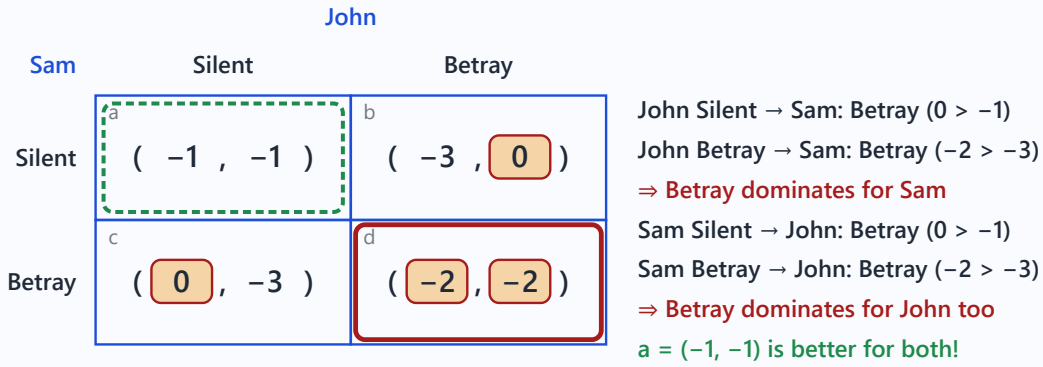


Figure 3.2: Best responses in the Prisoner's Dilemma. The orange boxes show each player's best response; the only cell with two marks, d = (Betray, Betray), is the Nash equilibrium (red frame). The green dashed frame, a = (Silent, Silent), is better for both players but is not an equilibrium.

Figure 3.2 shows the final state of the marking. Both of each player's marks lie in the Betray row or column. This shows that Betray is a dominant strategy for both players. A player with a dominant strategy does not need to predict the opponent, which makes the game very easy to solve.

3.2.3 A three-by-three example

Not every game has a dominant strategy. In the game below, the row player chooses among A, B, C and the column player among X, Y, Z. Each cell contains (u_1, u_2): first the row player's payoff, then the column player's.

	X	Y	Z
A	**(4, 3)**	(0, 2)	(1, 1)
B	(2, 1)	**(3, 3)**	(0, 2)
C	(1, 4)	(2, 0)	(3, 2)

Table 3.3: A three-by-three example game. The green cells are the two pure Nash equilibria.

Example 3.3

The row player's best responses (compare the first numbers in each column):

- If the column player plays X: A: 4, B: 2, C: 1 → best is A (4).
- If the column player plays Y: A: 0, B: 3, C: 2 → best is B (3).
- If the column player plays Z: A: 1, B: 0, C: 3 → best is C (3).

The column player's best responses (compare the second numbers in each row):

- If the row player plays A: X: 3, Y: 2, Z: 1 → best is X (3).
- If the row player plays B: X: 1, Y: 3, Z: 2 → best is Y (3).
- If the row player plays C: X: 4, Y: 0, Z: 2 → best is X (4).

Intersection: The cells in which both payoffs are marked are (A, X) = (4, 3) and (B, Y) = (3, 3). The game has two pure Nash equilibria.

Deviation test (A, X): If the row player switches to B they get 2, and to C they get 1; both are less than 4. If the column player switches to Y they get 2, and to Z they get 1; both are less than 3. Nobody gains by deviating ✓. The same check works for (B, Y): the row player drops to 2 or 0, the column player to 1 or 2 ✓.

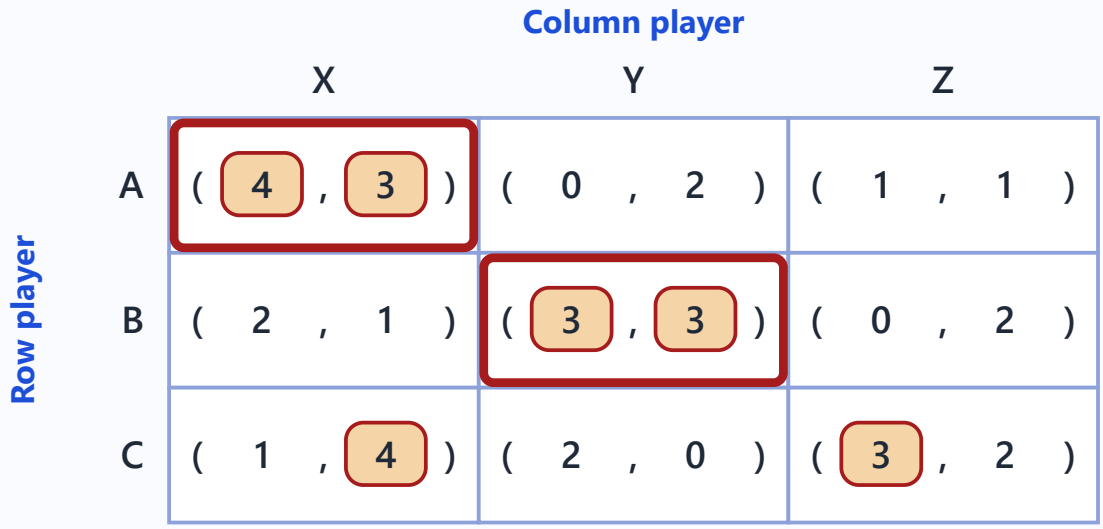


Figure 3.3: The result of marking in the three-by-three game. The row player has one mark in each column (4, 3, 3), the column player one in each row (3, 3, 4). The cells with two marks, (A, X) and (B, Y), are equilibria.

This example offers two important observations. First, in cell (C, Z) the row player's 3 is marked, but the column player can switch from Z to X and get 4 instead of 2. A single mark is not enough. Second, "large numbers" do not determine the equilibrium: the 4 in (C, X) is also one of the largest payoffs in the table for the column player, but the row player wants to switch to A there. An equilibrium is not a cell with large numbers; it is a cell that is a mutual best response.

It is also notable that the game has two equilibria. The theory alone does not say which equilibrium will be played. The row player prefers (A, X) (getting 4); the column player gets 3 in both equilibria and is therefore indifferent between them. Which one is chosen when there are several equilibria is the subject of Section 3.5.

3.2.4 When there are ties: the language-choice game

The marking method needs some care when there are ties. Ayşe and Bora are choosing a programming language for a joint project: Python, Java, or C. Each cell holds the payoffs (Ayşe, Bora); Ayşe is the row player and Bora the column player. The rule is the same: for Ayşe, underline the largest first number in each column; for Bora, underline the largest second number in each row. This is why the method is also called the **underlining method**.

Ayşe \ Bora	Python	Java	C
Python	**(4, 2)**	(1, 1)	(1, 0)
Java	(0, 1)	**(2, 4)**	(1, 2)
C	(1, 0)	(0, 2)	(1, 1)

Table 3.4: The language-choice game. Each cell shows (Ayşe, Bora).

Example 3.4

1. If Bora chooses Python, Ayşe looks only at the first numbers: 4, 0, 1 → the largest, 4 (Python), is underlined.
2. If Bora chooses Java: 1, 2, 0 → the largest, 2 (Java).
3. If Bora chooses C: 1, 1, 1 → all three are equal and all three are underlined. With a tie, every option is a best response; the argmax set is {Python, Java, C}.
4. If Ayşe chooses Python, Bora looks at the second numbers: 2, 1, 0 → 2 (Python).
5. If Ayşe chooses Java: 1, 4, 2 → 4 (Java).
6. If Ayşe chooses C: 0, 2, 1 → 2 (Java).
7. The cells where both numbers are underlined are (Python, Python) = (4, 2) and (Java, Java) = (2, 4). In (C, C) only the first number is underlined: Bora can switch to Java and get 2 instead of 1, so (C, C) is not an equilibrium.



Figure 3.4: Underlining in the language-choice game. In the C column Ayşe's three payoffs are equal (1, 1, 1), so all three are underlined. The cells with two underlines, (Python, Python) and (Java, Java), are the pure equilibria.

Marking only one option in a tie is a common mistake. In this example the mistake may look harmless. In another game, though, one of the tied cells may be exactly the cell where the equilibrium lies; a single mark would miss that equilibrium. The rule is clear: if the largest value occurs more than once, all of them are marked.

The same game can also be examined with the elimination of dominated strategies from Section 3.3. For Bora, Java strictly beats C in every row ($1 > 0$, $4 > 2$, $2 > 1$), so the C column can be deleted. In the two remaining columns, Python beats C for Ayşe ($4 > 1$, $1 > 0$), and the C row is deleted too. In the remaining 2×2 game no strategy is dominated; both equilibria remain. So elimination shrinks the

game but does not solve it, whereas underlining finds the two equilibria directly. The game also has a mixed equilibrium (Ayşe plays Python with probability 3/4, Bora plays Python with probability 1/5); equilibria of this kind are computed in Section 3.4.

3.2.5 Common mistakes in marking

Marking is a mechanical method, but it is easy to mix up which numbers are compared in which direction. Figure 3.5 shows the correct directions: because the row player can change only their row, they compare their options *within the same column*; the column player compares *within the same row*.

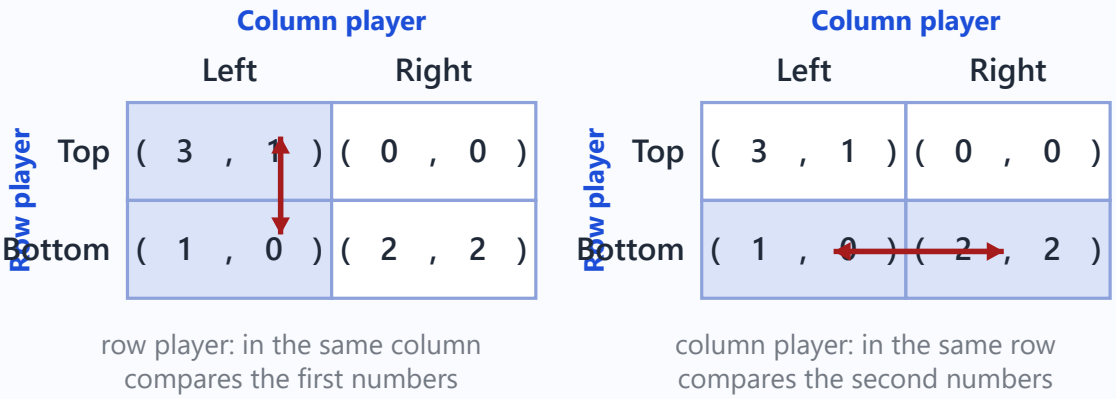


Figure 3.5: Directions of comparison. Left: the row player compares the first numbers (3 and 1) in the same column. Right: the column player compares the second numbers (0 and 2) in the same row.

Common mistake

- **Wrong direction:** Looking for the row player's best along a row. They can change only their row; look within the same column.
- **Wrong number:** Looking at the first numbers for the column player. The column player wants to make the second number larger.
- **A single mark in a tie:** If the largest value occurs twice, both are best responses and both are marked.
- **Thinking one mark is enough:** A cell marked for only one player is not an equilibrium; two marks are needed.
- **Looking for the biggest cell:** An equilibrium is not a cell with large numbers; it is a cell that is a mutual best response.
- **Skipping the check:** Run the deviation test on every candidate cell: does the player get more by switching to another option?

3.2.6 Three players: the group-project game

The definition is not limited to two players. Three students each say *Join* or *Skip* for a group project, simultaneously. If at least two join, the project succeeds and everyone (those who joined and those who did not) gets 3 points. Joining has a personal cost of 1. With three players and two strategies each, there are 2 × 2 × 2 = 8 profiles. Because the game is symmetric, it is enough to group the profiles by the number of joiners (k).

Joiners	Payoffs	Unilateral deviation	Equilibrium?
k = 3 (everyone)	3 − 1 = 2 each	If one drops out, the project still succeeds (k = 2) and the one who dropped out gets 3: 3 > 2	No
k = 2	joiners 2, the one who skipped 3	If a joiner drops out, the project fails: 0 < 2. If the one who skipped joins: 2 < 3	Yes (3 profiles)
k = 1	the joiner −1, the others 0	If the joiner drops out: 0 > −1	No
k = 0 (nobody)	0 each	Someone who joins alone gets −1: −1 < 0	Yes

Table 3.5: Deviation test for each profile in the group-project game.

The game has four pure equilibria: the three profiles in which exactly two students join, and the profile in which nobody joins. The test is the same in an n-player game: with the others held fixed, each player is checked separately. With three players a single "matrix" cannot be drawn, but the definition does not change at all.

The "nobody joins" equilibrium is interesting. Everyone gets 0, although if two students joined, the joiners would get 2 and the one who skipped would get 3. Nobody can break this equilibrium alone, because a student who joins alone does not save the project and gets −1. This situation is called a **coordination failure**. If two students decided to join together, things would improve, but a joint deviation lies outside the Nash definition. In this way the Nash equilibrium also explains how a group can "get stuck" in a bad situation.

3.2.7 Code trace: pure_nash()

The marking method can be written as a short Python function. The function takes the row player's payoff matrix `U1` and the column player's payoff matrix `U2` and returns the (row, column) indices of the pure equilibria.

```
def pure_nash(U1, U2):
    m, n, eq = len(U1), len(U1[0]), []
    for i in range(m):
        for j in range(n):
            br1 = max(U1[k][j] for k in range(m))
            br2 = max(U2[i][l] for l in range(n))
            if U1[i][j] == br1 and U2[i][j] == br2:
                eq.append((i, j))
    return eq
```

The code works line by line as follows:

- `m` is the number of rows and `n` the number of columns; `eq` is an empty list.
- Two nested loops visit every cell: `i` is the row index and `j` the column index.
- `br1` is the largest `U1` value in column `j`. It is the best payoff the row player can get against this column.
- `br2` is the largest `U2` value in row `i`. It is the best payoff the column player can get against this row.
- If both payoffs in the cell equal their maxima (that is, both marks are present), the cell is added to the list. Because equality (`==`) is used, all tied maxima are accepted; the tie problem resolves itself.

For the game in Table 3.3, `U1 = [[4, 0, 1], [2, 3, 0], [1, 2, 3]]` and `U2 = [[3, 2, 1], [1, 3, 2], [4, 0, 2]]`; indices 0, 1, 2 stand for A, B, C (or X, Y, Z) respectively. The column maxima are X: 4, Y: 3, Z: 3; the row maxima are A: 3, B: 3, C: 4. The trace table shows the nine cells in order.

(i, j)	Cell	U1[i][j] / br1	U2[i][j] / br2	Condition	eq
(0, 0)	(A, X)	4 / 4	3 / 3	true	[(0, 0)]
(0, 1)	(A, Y)	0 / 3	2 / 3	false	same
(0, 2)	(A, Z)	1 / 3	1 / 3	false	same
(1, 0)	(B, X)	2 / 4	1 / 3	false	same
(1, 1)	(B, Y)	3 / 3	3 / 3	true	[(0, 0), (1, 1)]
(1, 2)	(B, Z)	0 / 3	2 / 3	false	same
(2, 0)	(C, X)	1 / 4	4 / 4	false	same
(2, 1)	(C, Y)	2 / 3	0 / 4	false	same
(2, 2)	(C, Z)	3 / 3	2 / 4	false	same

Table 3.6: Trace of `pure_nash(U1, U2)`.

After the nine cells have been scanned, the function returns `[(0, 0), (1, 1)]`, that is, (A, X) and (B, Y). Note that in (C, X) only the second condition holds and in (C, Z) only the first; since `and` demands both, these cells are eliminated. This is the code counterpart of the rule "one mark is not enough".

In terms of complexity, this version computes in every cell one column maximum (m comparisons) and one row maximum (n comparisons). The total work is $O(mn \cdot (m + n))$. If the maxima are computed once before the loop and stored in a list, the work drops to $O(mn)$. For small matrices the difference is negligible; for a 1000 × 1000 matrix it is a factor of about 2000.

3.2.8 Continuous strategies: the Cournot duopoly

So far each player has had a handful of options. In some games a strategy is a number that can take infinitely many values. The oldest example is the two-firm market studied by Antoine Augustin Cournot in 1838; a market with two firms is called a **duopoly**.

Two firms produce the same good and choose their output quantities $y_1, y_2 \geq 0$ simultaneously. The market price depends on total output: $P = 12 - (y_1 + y_2)$. The more is produced, the lower the price. Production cost is zero. Firm 1's profit is quantity times price:

$$u_1(y_1, y_2) = y_1 \cdot (12 - y_1 - y_2)$$

No matrix can be drawn here, because the number of rows is infinite. But the idea of best response applies unchanged: while firm 2's quantity y_2 is held fixed, we look for the y_1 that maximizes firm 1's profit. When the choice is continuous, the argmax is not read off a table; it is found with the **derivative**.

The derivative is the slope of a curve at each point. At the top of the profit curve the slope is zero: to the left of the peak the slope is positive (profit is rising), and to the right it is negative (profit is falling). Three rules are enough here: the derivative of y^2 is $2y$; the derivative of $c \cdot y$ is c; the derivative of a constant is 0. The derivative of a sum is the sum of the derivatives.

Example 3.5

1. Write the profit explicitly. $u_1 = 12y_1 - y_1^2 - y_1y_2$. For firm 1, y_2 is a constant.

2. Take the derivative. The derivative of $12y_1$ is 12, that of y_1^2 is $2y_1$, and that of y_1y_2 is y_2. The result: $12 - 2y_1 - y_2$.

3. Set it to zero. $12 - 2y_1 - y_2 = 0 \Rightarrow y_1 = (12 - y_2)/2$.

4. Peak or valley? The derivative of the derivative (the second derivative) is −2, which is negative; the slope is decreasing, so the point is a peak. A numerical check works too: for $y_2 = 4$, $u_1(4) = 4 \cdot 4 = 16$, and at the neighbors $u_1(3) = 3 \cdot 5 = 15$ and $u_1(5) = 5 \cdot 3 = 15$; both are less than 16 ✓.

5. Do not forget the boundary. Quantity cannot be negative. If $y_2 \geq 12$, the formula gives zero or a negative number; in that case the price is already zero and the best choice is not to produce. The full best response is:

$$BR_1(y_2) = \max\{0, (12 - y_2)/2\}$$

6. Find the equilibrium. Because the game is symmetric, $BR_2(y_1) = \max\{0, (12 - y_1)/2\}$. An equilibrium is a point where the two firms are simultaneously best-responding to each other: $y_1 = (12 - y_2)/2$ and $y_2 = (12 - y_1)/2$. Setting $y_1 = y_2 = y$ by symmetry gives $2y = 12 - y$, so $y = 4$. The price is $12 - 8 = 4$, and each firm's profit is $4 \cdot 4 = 16$.

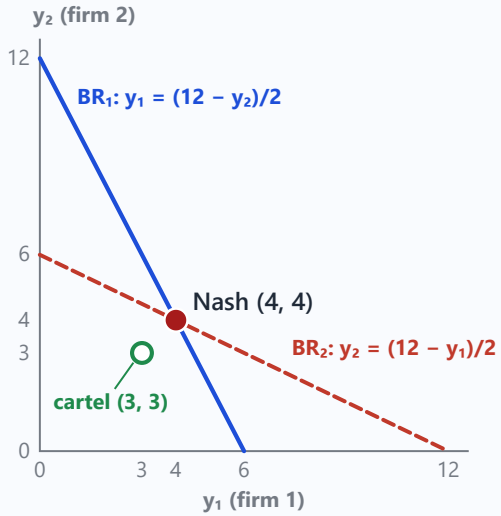


Figure 3.6: Best-response lines in the Cournot duopoly. The solid blue line is BR_1 and the dashed red line is BR_2. Their intersection (4, 4) is the Nash equilibrium; the green point (3, 3) is the cartel output and lies on neither line.

Figure 3.6 draws the two best-response lines. The horizontal axis is firm 1's quantity and the vertical axis is firm 2's. The BR_1 line shows firm 1's best quantity for each quantity of firm 2; it starts at 6 for $y_2 = 0$ and falls to 0 at $y_2 = 12$. The equilibrium is the single point where the two lines cross. The slope of each line is negative: the more the rival produces, the less the firm's best quantity.



Figure 3.7: Interactive Cournot best response. The slider changes the rival's quantity y_2; on the left you see firm 1's profit curve $u_1(y_1) = y_1(12 - y_1 - y_2)$ and its peak, on the right the corresponding point on the best-response line. The default is $y_2 = 4$, and the best response is 4.

Why is the cartel not an equilibrium? If the two firms agreed to make total profit as large as possible, they would maximize $Y(12 - Y)$ over total quantity Y. The derivative $12 - 2Y = 0$ gives $Y = 6$; each firm produces 3, the price is 6, and each firm earns $3 \cdot 6 = 18$. That is more than the 16 at equilibrium. But the agreement is not self-enforcing. While the rival produces 3, firm 1's best response is $BR_1(3) = (12 - 3)/2 = 4.5$. The deviating firm earns $4.5 \cdot (12 - 4.5 - 3) = 4.5 \cdot 4.5 = 20.25$, while the firm that sticks to the agreement sees its profit fall to $3 \cdot 4.5 = 13.5$. Each firm has an incentive to deviate. This is a continuous-strategy version of the Prisoner's Dilemma.

How can the equilibrium be found? We do not have to assume that the firms compute the equilibrium. If the firms respond to each other with best responses in turn, the quantities approach the equilibrium. Suppose firm 1 starts alone with 6 (the best response when the rival produces 0). Firm 2 answers with $(12 - 6)/2 = 3$. Firm 1 answers 3 with 4.5, and the process continues in this way.

Step	Updates	Computation	(y_1, y_2)
0	start	firm 1 alone	(6, 0)
1	firm 2	(12 − 6)/2 = 3	(6, 3)
2	firm 1	(12 − 3)/2 = 4.5	(4.5, 3)
3	firm 2	(12 − 4.5)/2 = 3.75	(4.5, 3.75)
4	firm 1	(12 − 3.75)/2 = 4.125	(4.125, 3.75)
5	firm 2	(12 − 4.125)/2 = 3.9375	(4.125, 3.9375)

Table 3.7: Sequential best responses in Cournot. In each row only one firm updates its quantity.

At each step, the distance of the updated quantity from the equilibrium is halved: 2 for 6, 1 for 3, 0.5 for 4.5, 0.25 for 3.75, 0.125 for 4.125. The reason is that the slope of the best-response line is −1/2. Such a "learning process" does not converge to the equilibrium in every game. Because it does here, the Cournot equilibrium is also a plausible prediction from a behavioral point of view.

3.2.9 Bertrand: competing on price

If the firms choose prices instead of quantities, the outcome changes surprisingly. This model bears the name of Joseph Bertrand. Two firms sell the same good; the unit cost is $c = 2$ TL and the market is 100 units. The firms choose prices simultaneously. The cheaper firm takes the whole market; if prices are equal, the market is split in half. A firm's profit is (price − cost) × quantity sold.

Example 3.6

1. **Is (5, 5) an equilibrium?** Each firm sells 50 units and earns $(5 - 2) \cdot 50 = 150$. If firm 1 cuts its price to 4.99, it takes the whole market: $2.99 \cdot 100 \approx 299$. The deviation nearly doubles the profit; (5, 5) is not an equilibrium.
2. **Any common price above c:** If the common price is $p > 2$, each firm earns $(p - 2) \cdot 50$; a firm that undercuts slightly earns about $(p - 2) \cdot 100$. None of these is an equilibrium.
3. **Different prices:** If the cheaper firm's price is above c, it can raise its price to just below the rival's, still take the whole market, and earn more. If the cheaper firm's price is exactly c, its profit is 0; it can raise its price to somewhere between c and the rival's price and make a profit. No profile with different prices is an equilibrium.
4. **Below c:** A firm that cuts its price below 2 loses money on every unit sold; this is never a best response.
5. **(2, 2):** Both firms earn 0. A firm that goes lower makes a loss, and one that goes higher sells nothing and still gets 0. No deviation increases profit. The only Nash equilibrium is $p_1 = p_2 = c = 2$, with profit 0.



Figure 3.8: The chain of price cuts in Bertrand. Because undercutting slightly is profitable from any common price above c, the price falls step by step to $p = c = 2$.

Means of competition	Equilibrium	Firm profit
Cournot (quantity), P = 12 − Y, cost 0	y = 4 each, price 4	16 each
Bertrand (price), cost c	p = c	0

Table 3.8: Comparison of Cournot and Bertrand.

The difference between the two models comes from what a deviation yields. In Cournot, when a firm increases its quantity the price falls, and that fall also reduces its own profit; the deviation brakes itself. In Bertrand, a small price cut wins the entire market; the reward for deviating is very large. So even two firms push the price down to cost. This result is called the **Bertrand paradox**: the intuition that "with two firms profit stays high" is wrong under price competition. In real markets profit is protected only by factors such as product differentiation, capacity constraints, or repeated interaction.

Common mistake

The conclusion "Bertrand's only equilibrium is p = c" rests on the assumption that price is continuous: arbitrarily small cuts such as 4.99 are possible. If prices can be chosen in steps of 1 TL, prices just above c can also be equilibria. In this example, at (3, 3) each firm earns $1 \cdot 50 = 50$; a firm that goes down to 2 earns 0, and one that goes up sells nothing, so the deviation is unprofitable. At (4, 4) each firm earns 100; a firm that goes down to 3 earns $1 \cdot 100 = 100$, so the deviation does not increase profit. At (5, 5), however, a firm that goes down to 4 earns 200, so the deviation is profitable. Conclusion: with 1 TL steps the equilibria are (2, 2), (3, 3), and (4, 4).

3.3 Elimination of dominated strategies

Best-response marking looks for the equilibrium directly. Elimination of dominated strategies works from the opposite direction: it finds the strategies that "no rational player would play" and removes them from the game, shrinking it. Sometimes a single cell remains, and the game is solved completely this way.

3.3.1 Dominant and dominated strategies

The strongest way to compare two strategies is to compare them against *every* possible choice of the opponent. If one strategy is better than another whatever the opponent does, there is no reason ever to play the second.

Definition 3.4

Let s_i' and s_i be two strategies of player i.

- **Strict dominance:** If $u_i(s_i', s_{-i}) > u_i(s_i, s_{-i})$ for every s_{-i}, then s_i' strictly dominates s_i.
- **Weak dominance:** If $\geq$ holds for every s_{-i} and $>$ holds for at least one s_{-i}, then s_i' weakly dominates s_i.

A strategy that is dominated is called a **dominated strategy**. A strategy that strictly dominates all of the player's other strategies is called a **dominant strategy**.

The word "every" in the definition is critical. For strict dominance, the comparison is made separately for each choice of the opponent, and a strict inequality is needed in each. "Better on average" or "better in most cases" is not enough.

A rational player never plays a strictly dominated strategy. The reason is clear: whatever the player thinks about the opponent, the dominating strategy earns more. A strictly dominated strategy is not a best response to any choice of the opponent.

For a player with a dominant strategy the game is easy: there is no need to predict the opponent. If everyone has a dominant strategy, the profile in which everyone plays their dominant strategy is a Nash equilibrium; it is called a **dominant-strategy equilibrium**. In the Prisoner's Dilemma, Betray is a dominant strategy for both players, and (Betray, Betray) is a dominant-strategy equilibrium. But not every game has a dominant strategy. In Rock-Paper-Scissors (RPS) each move loses to another; no move beats the others in every case.

Even when there is no dominant strategy, there may be individual dominated strategies. Deleting them and searching again in the smaller game yields a method.

Definition 3.5

Iterated elimination of strictly dominated strategies (IESDS): a strictly dominated strategy of any player is deleted; in the smaller game, strictly dominated strategies are searched for again. The process stops when there is no strategy left to delete.

The method assumes more than that the players are rational. Everyone's rationality is known to everyone, that is known to everyone, and the chain continues forever. This assumption is called **common knowledge**. A strategy deleted in the first round follows only from the player's own rationality. A strategy deleted in the second round requires the player to know that the opponent is rational: "my opponent will not play that dominated strategy, so I do not need to consider that column". Each new round rests on one more level of knowledge.

Theorem 3.2 (Safety of strict elimination)

In iterated elimination of strictly dominated strategies, (a) no Nash equilibrium is deleted, and (b) the result does not depend on the order of deletion. In particular, if a single cell remains, that cell is the game's only Nash equilibrium.

Proof. The idea of part (a) is as follows. In a Nash equilibrium, each player's strategy is a best response to the others' equilibrium strategies. A strictly dominated strategy cannot be a best response to any choice of the opponent, because the dominating strategy earns more against that choice too. So a strictly dominated strategy appears in no equilibrium, and deleting it destroys no equilibrium. In the smaller game after the deletion, the equilibrium strategies are still best responses, because no option that beats them has been added; the options have only been reduced. The same reasoning repeats in every round. The proof of part (b) is longer and is not given here (Osborne, 2004, Chapter 12). □

For weak dominance there is no such guarantee. A weakly dominated strategy yields the *same* payoff as the dominating one in some cases; in those cases it may be a best response and may appear in an equilibrium. When it is deleted, some equilibria may disappear, and the result may depend on the order of deletion (Section 3.3.7).

A familiar example shows the difference. In the **2/3-of-the-average game**, everyone writes a number between 0 and 90; the number closest to 2/3 of the average wins. Because the average can be at most 90, the target is at most 60. A guess above 60 is never better than writing 60, but in some cases (for example, when both guesses are far from the target and both lose) it gives an equal result. So deleting the numbers above 60 relies on weak dominance, not strict dominance.

3.3.2 Back to the Prisoner's Dilemma

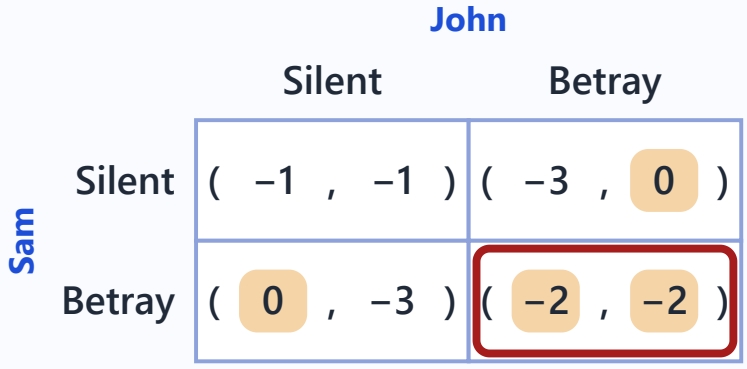


Figure 3.9: The Prisoner's Dilemma, payoff = −years. Each player's best responses are marked; both marks are in the Betray row and column.

Should Sam stay silent or betray? The theory's answer is clear: Betray is the dominant strategy, because *0 > −1* if John is silent and *−2 > −3* if he betrays. The only Nash equilibrium is (Betray, Betray) = (−2, −2), that is, two years each (Figure 3.9).

Real behavior differs from the theory. In the one-shot game, people cooperate much more often than the theory predicts. According to a meta-analysis combining experiments from 1958 to 1992, roughly half of the participants cooperate, and allowing talk before the game markedly raises this rate (Sally, 1995). Feelings of trust, fairness, and reciprocity also influence the choice. This does not show that the theory is wrong. When the payoff is just years in prison, other things people care about (guilt, reputation) fall outside the table. The theory is a clear point of comparison that shows what would happen without these factors.

The repeated game. When the game is played repeatedly against the same opponent, the situation changes. Let the computer play Tit-for-Tat: it stays silent in the first round, then copies the opponent's previous choice. The total cost of Sam's different 10-round plans is as follows.

Sam's plan	What happens round by round?	Sam's total years
Always Silent	John also stays silent throughout; 1 year per round.	10 × 1 = 10
Always Betray	Round 1: John is silent, so Sam gets 0. In the next 9 rounds John betrays too: 2 years each.	0 + 9 × 2 = 18
Silent for 9 rounds, Betray in the last	1 year in each of the first 9 rounds; in round 10 John is silent, Sam gets 0, and no punishment can follow.	9 × 1 + 0 = 9 (minimum)

Table 3.9: Sam's 10-round plans when John plays Tit-for-Tat.

Betray, which dominates in the one-shot game, is expensive against Tit-for-Tat: every betrayal is punished in the next round. Only in the last round can no punishment follow, because there is no next round. So the best plan is to stay silent for nine rounds and betray in the last. Trying all $2^{10} = 1024$

plans by computer gives the same result: the minimum total is 9 years. This "last-round effect" is the subject of **backward induction**, which reasons from the end of a game tree back to the start.

3.3.3 Iterated elimination: a three-by-three example

In the game below, neither player has a dominant strategy, but iterated elimination reduces the game to a single cell.

Row player \ Column player	X	Y	Z
A	(3, 2)	(3, 3)	(1, 4)
B	(3, 2)	(0, 4)	(5, 3)
C	(2, 1)	(2, 0)	(6, 6)

Figure 3.10: An example of iterated elimination. The row player chooses A, B, C; the column player chooses X, Y, Z; each cell shows (u_1, u_2).

Example 3.7

1. **Start.** Rows are compared using the first numbers. No row strictly dominates another; for example, A and B are equal in column X (3 = 3). Since no progress can be made with rows, we look at the column player.
2. **X is eliminated.** Columns are compared using the second numbers. Z beats X in every row: 4 > 2 in A, 3 > 2 in B, 6 > 1 in C. X is strictly dominated and is deleted.
3. **B is eliminated.** Now only columns Y and Z remain. C beats B in these two columns: 2 > 0 in Y, 6 > 5 in Z. (While X was present, this comparison was invalid: in X, B gets 3 and C gets 2.)
4. **Y is eliminated.** The remaining rows are A and C. Z beats Y in both rows: 4 > 3 in A, 6 > 0 in C.
5. **A is eliminated.** Only column Z is left. C beats A: 6 > 1.
6. **Result.** A single cell remains: (C, Z) = (6, 6). By Theorem 3.2, this cell is the game's only Nash equilibrium. Best-response marking gives the same result.

Step	Remaining rows	Remaining columns	Deleted	Reason
1	A, B, C	X, Y, Z	X	Z > X: 4 > 2, 3 > 2, 6 > 1
2	A, B, C	Y, Z	B	C > B: 2 > 0, 6 > 5
3	A, C	Y, Z	Y	Z > Y: 4 > 3, 6 > 0
4	A, C	Z	A	C > A: 6 > 1
5	C	Z	—	(C, Z) = (6, 6)

Table 3.10: The elimination order and the comparison at each step.

3.3.4 One elimination opens the way for the next

The power of iterated elimination is that each deletion opens the door to new deletions. In the game below there is initially a single strict dominance; all the others appear after it. Rows are compared using only the first numbers, columns using only the second numbers.

	Left	Center	Right
Top	(1, 5)	(5, 6)	(6, 2)
Middle	**(3, 5)**	(6, 0)	(2, 2)
Bottom	(2, 2)	(0, 2)	(1, 5)

Table 3.11: The elimination-chain game. Each cell shows the payoffs (row, column).

Example 3.8

1. **Bottom is deleted.** Middle beats Bottom in every column: 3 > 2, 6 > 0, 2 > 1. There is no other strict dominance: for example, Left cannot beat Right, because in the Bottom row Right gives 5 and Left gives 2.
2. **Right is deleted.** Once Bottom is gone, Left beats Right in the two remaining rows: 5 > 2 in Top, 5 > 2 in Middle. This would not have happened while Bottom was present.
3. **Top is deleted.** Once Right is gone, Middle beats Top in the two remaining columns: 3 > 1 in Left, 6 > 5 in Center. While Right was present, Top was saved there by 6 > 2.
4. **Center is deleted.** Only the Middle row is left: Left gives 5, Center gives 0. Center is deleted.
5. **Result.** The remaining (Middle, Left) = (3, 5) is the only Nash equilibrium.



Figure 3.11: The elimination chain in its final state. First Bottom, then Right, then the Top row, and finally the Center column have been deleted (red lines). The remaining (Middle, Left) is the only equilibrium.

Two lessons follow from the example. First, the order of deletion is a logical chain: deleting Top depends on deleting Right, and deleting Right depends on deleting Bottom. Each step adds one more level to the knowledge of rationality assumed by the previous one. Second, the equilibrium is not the best outcome for everyone. (Top, Center) = (5, 6) is better for both players than (3, 5). But it is not an equilibrium: while the column player plays Center, the row player can switch to Middle and get 6 instead of 5.

3.3.5 Code trace: eliminate()

Iterated elimination is also written as a short function. The function keeps the remaining row indices in `R` and the remaining column indices in `S`.

```
def eliminate(U1, U2):
    R, S = [0, 1, 2], [0, 1, 2]   # remaining
    while True:
        x = [r for r in R for k in R if
             all(U1[k][c] > U1[r][c] for c in S)]
        y = [c for c in S for d in S if
             all(U2[r][d] > U2[r][c] for r in R)]
        if x: R.remove(x[0])      # drop a row
        elif y: S.remove(y[0])    # drop a column
        else: return R, S
```

The logic of the code is as follows:

- `x` is the list of remaining rows that are strictly dominated by another remaining row `k`. `all(U1[k][c] > U1[r][c] for c in S)` checks that row `k` strictly beats row `r` in *every* remaining column. Deleted columns are not in `S`, so they do not enter the comparison.
- `y` makes the same check for columns using `U2`: does column `d` beat column `c` in every remaining row?
- Row deletion is tried first: if `x` is not empty, the first dominated row is deleted. Otherwise columns are tried. If both are empty, the remaining sets are returned.
- The `while True` loop starts again from the top after every deletion; this way the new dominations opened by a deletion are found in the next round.

Called with the matrices of the three-by-three elimination example, the trace is as follows:

Round	R	S	x	y	Action
1	[0, 1, 2]	[0, 1, 2]	[]	[0]	`S.remove(0)`: X is deleted (Z dominates)
2	[0, 1, 2]	[1, 2]	[1]	[]	`R.remove(1)`: B is deleted (C dominates)
3	[0, 2]	[1, 2]	[]	[1]	`S.remove(1)`: Y is deleted (Z dominates)
4	[0, 2]	[2]	[0]	[]	`R.remove(0)`: A is deleted (C dominates)
5	[2]	[2]	[]	[]	`return [2], [2]`

Table 3.12: Trace of `eliminate(U1, U2)` (0, 1, 2 = A, B, C and X, Y, Z).

The function returns `([2], [2])`, that is, only (C, Z) remains. Four deletions removed X, B, Y, A in that order. It is important that `y` comes out empty in round 2: at that moment, with three rows remaining, Z cannot beat Y, because in row B, Y gives 4 and Z gives 3. After B is deleted (round 3), Z beats Y. This is the code counterpart of the observation in Section 3.3.4 that "one elimination opens the way for the next".

In each round `x` makes at most $m^2 \cdot n$ comparisons and `y` at most $n^2 \cdot m$, and there are at most $m + n$ rounds. The total work is polynomial; for small games the computation finishes instantly.

3.3.6 Domination by a mixed strategy

There may be no single pure strategy that beats a given strategy, yet a mixture of two strategies may beat it. The table below gives only the row player's payoffs (u_1), because rows are compared using these numbers alone.

u_1	Left	Right
Top	4	0
Middle	1	1
Bottom	0	4

Table 3.13: The row player's payoffs.

No pure row dominates Middle: it is better than Top in the Right column (1 > 0) and better than Bottom in the Left column (1 > 0). But if the row player flips a coin and plays Top with probability one half and Bottom with probability one half, the **expected payoff** in each column is:

- Against Left: $\frac{1}{2} \cdot 4 + \frac{1}{2} \cdot 0 = 2$.
- Against Right: $\frac{1}{2} \cdot 0 + \frac{1}{2} \cdot 4 = 2$.

In both columns 2 > 1. So the ½–½ mixture strictly dominates Middle, and Middle can be deleted. Expected payoff is treated in detail in Section 3.4.1; here it is enough to read it as "the average payoff per round over many repetitions".

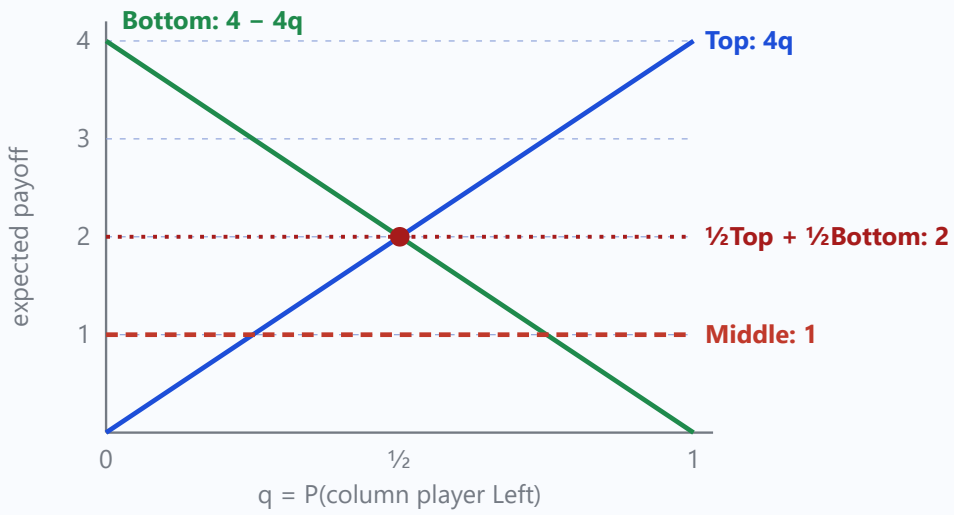


Figure 3.12: Domination by a mixed strategy. The horizontal axis is the probability q that the column player plays Left. The expected payoff of Top is $4q$, that of Bottom $4 - 4q$, and that of Middle is always 1. ½Top + ½Bottom gives 2 for every q.

Figure 3.12 takes the result one step further. Suppose the row player believes the column player will play Left with probability q. The expected payoff of Top is $4q$, that of Bottom $4(1 - q)$. The larger of these two lines (the upper envelope) is at least 2; its lowest point is the intersection at $q = \frac{1}{2}$. The line for Middle is 1 everywhere and stays entirely below the upper envelope. Conclusion: Middle is a best response to *no* belief about the column player. In two-player games, being strictly dominated by a mixed strategy and "being a best response to no belief" are the same thing.

Common mistake

"No single pure strategy beats it, so it cannot be deleted." Mixtures also enter the comparison. When eliminating, try pure strategies first, then simple mixtures such as ½–½. In particular, strategies that lie "in the middle" of two extreme strategies and are never best are deleted this way.

3.3.7 The trap of weak domination

A small game shows why deleting weakly dominated strategies is dangerous.

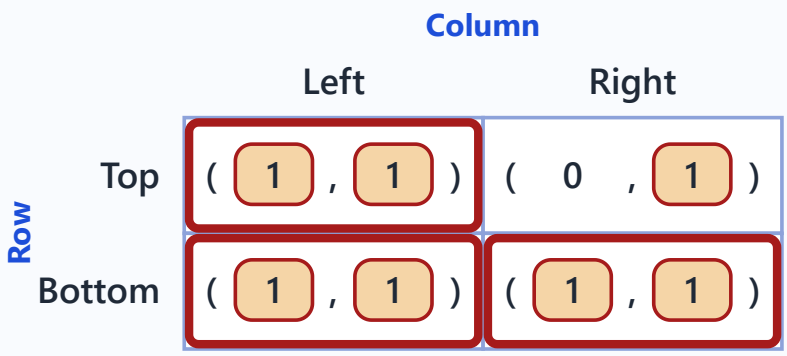


Figure 3.13: An example of weak domination. The column player gets 1 in every cell. The three cells with two marks are the three pure equilibria.

Example 3.9

1. For the row player, Bottom weakly dominates Top: 1 = 1 in Left, 1 > 0 in Right. Because of the tie in Left, the domination is not strict.
2. Since the column player gets 1 in every cell, they are indifferent in every case; every column is a best response to every row.
3. Pure equilibria: (Top, Left), (Bottom, Left), (Bottom, Right). (Top, Right) is not an equilibrium, because the row player can switch to Bottom and get 1 instead of 0. Three equilibria in all.
4. If Top is deleted, only the Bottom row remains and two equilibria remain: (Bottom, Left) and (Bottom, Right). The equilibrium (Top, Left) has disappeared.

Strict domination is safe; weak domination can cost equilibria. When there are several weak dominations, different cells may remain depending on which is deleted first. For this reason only strictly dominated strategies are deleted when searching for equilibria.

Strict elimination does not reduce every game to a single cell either. In the **Battle of the Sexes**, two people decide separately whether to go to the Opera or to Football in the evening. One likes the Opera more, the other prefers Football; but both would rather be together than go to different places. In this game no strategy is dominated and elimination deletes nothing. The equilibria are found by best-response marking and with mixed strategies.

3.3.8 Common mistakes in elimination

Common mistake

- **Wrong numbers:** Comparing two rows using the column player's payoffs. Row ↔ first number, column ↔ second number.
- **Thinking "mostly larger" is enough:** For strict domination it must be larger in every remaining column; a single tie or smaller value stops the elimination.
- **Counting what has been deleted:** A deleted column no longer enters the comparison. In an elimination chain, a strategy (for example Top) can be eliminated only after another strategy (Right) has been deleted.
- **Mistaking weak for strict:** If there is a tie, the domination is weak; deleting may lose equilibria and the result may depend on the order.
- **Confusing dominant with dominated:** A dominant strategy beats all the others; a dominated strategy loses to at least one. Elimination also works without a dominant strategy.
- **Assuming everything left over is an equilibrium:** If elimination leaves several cells (for example 2 × 2 remains in the language-choice game), the equilibria are found in the remaining small game by underlining.

3.4 Mixed strategies

In some games no cell is a mutual best response. In Matching Pennies, two players simultaneously choose a side of a coin; if the sides match, the row player wins, and if they differ, the column player wins. If the row player says Heads, the column player wants to say Tails; if the column player says Tails, the row player wants to switch to Tails, and the cycle goes on. In such games players want to be unpredictable. The idea of "being unpredictable" is made precise with probability.

3.4.1 Mixed strategies and the indifference principle

Two basic probability concepts are needed. A **probability distribution** assigns to each possible outcome a number between 0 and 1, and these numbers sum to 1. On a fair die each face has probability 1/6. The **expected value** of a random number X is found by multiplying each possible value by its probability and adding:

$$E[X] = \Sigma_x \, x \cdot P(X = x)$$

For a fair die, $E[X] = (1 + 2 + 3 + 4 + 5 + 6) \cdot 1/6 = 21/6 = 3.5$. The expected value is not "the most likely outcome"; a die never shows 3.5. The expected value says what the average per round would be if the same experiment were repeated many times.

Definition 3.6

A **mixed strategy** σ_i of player i is a probability distribution over their pure strategies: each pure strategy is given a nonnegative probability, and the probabilities sum to 1. A mixed strategy that gives probability 1 to a single pure strategy is that pure strategy itself.

The **support** of a mixed strategy is the set of pure strategies that are given positive probability.

When players play mixed strategies, the payoff is the **expected payoff**: the payoff of each cell is multiplied by the probability of that cell occurring, and the products are added. The players' dice are independent of each other; hence the probability of cell (row r, column c) is $\sigma_1(r) \cdot \sigma_2(c)$.

For example, in Rock-Paper-Scissors, (1/3, 1/3, 1/3) means "roll a die; play Rock on 1–2, Paper on 3–4, Scissors on 5–6". In 2 × 2 games the notation gets shorter: p is the probability that the row player plays the first row, and q the probability that the column player plays the first column; the probabilities of the other options are $1 - p$ and $1 - q$.

The expected payoff of a pure strategy against a mixing opponent is the payoffs of that row weighted by the opponent's probabilities. Let the opponent play (1/2 Rock, 1/4 Paper, 1/4 Scissors) in Rock-Paper-Scissors. Let a win be +1, a loss −1, and a draw 0:

- Rock: $\frac{1}{2} \cdot 0 + \frac{1}{4} \cdot (-1) + \frac{1}{4} \cdot (+1) = 0$.
- Paper: $\frac{1}{2} \cdot (+1) + \frac{1}{4} \cdot 0 + \frac{1}{4} \cdot (-1) = \frac{1}{4}$.
- Scissors: $\frac{1}{2} \cdot (-1) + \frac{1}{4} \cdot (+1) + \frac{1}{4} \cdot 0 = -\frac{1}{4}$.

The best response to this opponent is Paper, which earns an average of ¼ point per round. Expected value and argmax together give the best response to a mixing opponent. If the opponent played (1/3, 1/3, 1/3), every move would earn 0 and no move would be better than another. This is exactly the equilibrium of RPS: both players play (1/3, 1/3, 1/3) and every action earns 0. Any mixture other than 1/3–1/3–1/3 can be exploited by the opponent; the computation just above shows how a player with a slight bias toward Rock is exploited with Paper.

Definition 3.7

A profile $\sigma^* = (\sigma_1^*, \ldots, \sigma_n^*)$ of mixed strategies is a **mixed Nash equilibrium** if and only if $u_i(\sigma_i^*, \sigma_{-i}^*) \geq u_i(\sigma_i, \sigma_{-i}^*)$ for every player i and every mixed strategy σ_i. Here u_i is the expected payoff.

The definition is the same as in the pure case; only the strategies and payoffs have been extended. The key to computing a mixed equilibrium is the following principle.

Theorem 3.3 (Indifference principle)

In a mixed Nash equilibrium, each player gets the same expected payoff from every pure strategy in their support, against the others' equilibrium mixtures. Pure strategies outside the support do not give more than this.

Proof. The expected payoff of player i's mixed strategy is the probability-weighted average of the expected payoffs of the pure strategies in the support. Let a and b be two strategies in the support, and let a give strictly more than b. Then if the probability of b is shifted to a, the average rises, because the weight of a smaller value falls and that of a larger value rises. This means there is a profitable deviation at the equilibrium, which contradicts the definition of equilibrium. So all strategies in the support give equal payoffs. If a strategy outside the support gave more than this common value, the player could again increase their payoff by shifting all the probability to it; this is a contradiction as well. □

The principle has a surprising consequence. For the row player to be willing to play their mixed strategy, they must be indifferent between their two options; this indifference depends on the column player's q. So the row player's indifference equation determines the column player's q. Conversely, the column player's indifference equation determines the row player's p. The rule is: **the row player's mixture is computed from the column player's payoffs (and vice versa).**

A mixed strategy has two interpretations. The first is literally rolling a die: in a penalty kick the kicker picks a side at random in their head. The second is the population interpretation: in a large population the probabilities are behavior frequencies, for example the share of penalties kicked to the left. Even if a player does not roll a die, their move is uncertain in the opponent's eyes; probability measures this uncertainty.

3.4.2 The inspection game step by step

The **inspection game** is a typical example of a game with no pure equilibrium. The row player is a worker and chooses Work or Shirk. The column player is a manager and chooses Inspect or Don't inspect. The numbers are: wage 4, the worker's cost of effort 2, the output of a working worker 6, the manager's cost of inspection 1. A shirking worker who is caught during an inspection gets no wage.

Each cell is derived from these numbers. In the Work–Inspect cell the worker subtracts the cost of effort from the wage: $4 - 2 = 2$. The manager subtracts the wage and the inspection cost from the output: $6 - 4 - 1 = 1$. In Work–Don't inspect the worker again gets 2 and the manager gets $6 - 4 = 2$. In Shirk–Inspect the worker is caught: no wage, no effort, so 0; the manager gets no output, pays no wage, and pays only the inspection cost: −1. In Shirk–Don't inspect the worker gets the wage without working: 4; the manager pays the wage without getting output: −4.

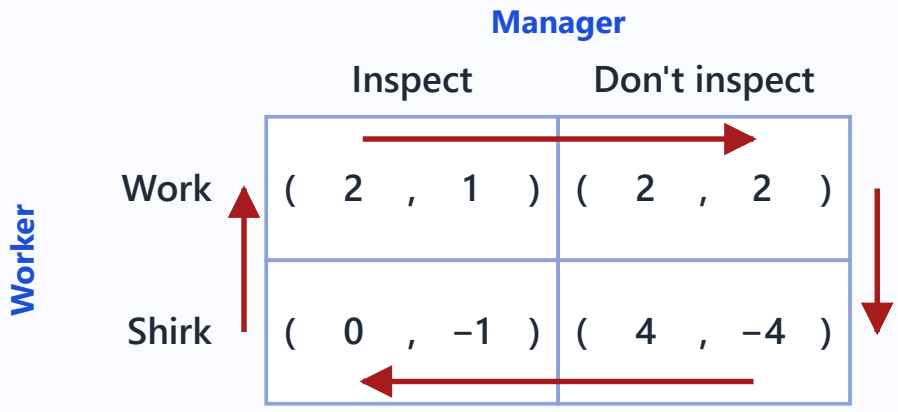


Figure 3.14: The inspection game; each cell shows (worker, manager). The arrows show a player's profitable deviation from each cell. The four arrows form a cycle: there is no pure Nash equilibrium.

Let us follow the arrows in Figure 3.14. If the worker works, the manager stops inspecting ($2 > 1$). If the manager does not inspect, the worker shirks ($4 > 2$). If the worker shirks, the manager inspects ($-1 > -4$). If the manager inspects, the worker works ($2 > 0$). In every cell someone wants to deviate; the best responses form a cycle. So we look for a mixed equilibrium.

Example 3.10

Let the worker play Work with probability p and the manager play Inspect with probability q.

1. **The worker's two options.** The expected payoff of Work is $E[\text{Work}] = 2q + 2(1 - q) = 2$; it is 2 whatever the manager does. The expected payoff of Shirk is $E[\text{Shirk}] = 0 \cdot q + 4(1 - q) = 4 - 4q$.
2. **The worker must be indifferent.** $2 = 4 - 4q \Rightarrow 4q = 2 \Rightarrow q^* = 1/2$. The manager must inspect half the time. If they inspect less, the worker always shirks; if they inspect more, the worker always works and the inspection is wasted money.
3. **The manager's two options.** $E[\text{Inspect}] = 1 \cdot p + (-1)(1 - p) = 2p - 1$; $E[\text{Don't inspect}] = 2p + (-4)(1 - p) = 6p - 4$.
4. **The manager must be indifferent.** $2p - 1 = 6p - 4 \Rightarrow 3 = 4p \Rightarrow p^* = 3/4$.
5. **Check.** Both probabilities lie between 0 and 1 ✓. Expected payoffs: the worker gets 2 (both options give 2), the manager gets $2 \cdot \tfrac{3}{4} - 1 = \tfrac{1}{2}$.

The equilibrium is $(p^*, q^*) = (3/4, 1/2)$. The worker works three quarters of the time, and the manager inspects half the time.

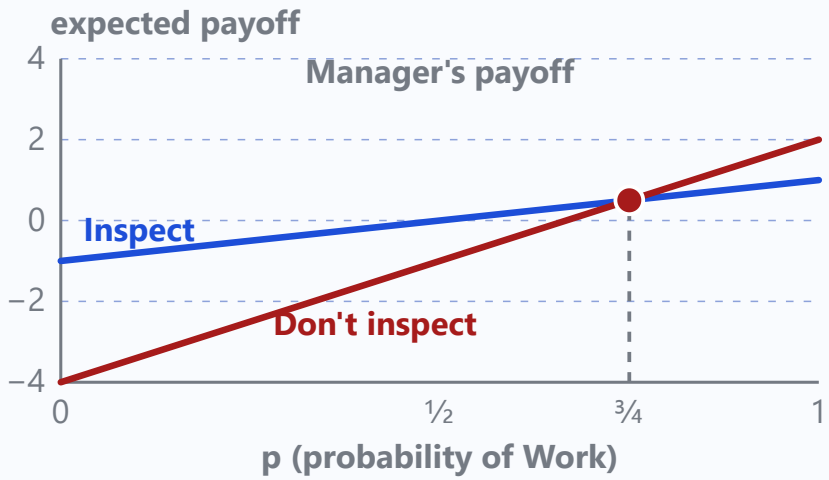


Figure 3.15: The manager's expected payoff. The blue line is Inspect ($2p - 1$), the red line is Don't inspect ($6p - 4$). The two lines cross at $p = 3/4$; if the worker works with this probability, the manager is indifferent.

Figure 3.15 shows the manager's situation. When $p < 3/4$, the blue line is on top: if the worker shirks often, it pays to inspect. When $p > 3/4$, the red line is on top: if the worker mostly works, inspection is unnecessary. Only at $p = 3/4$ is the manager indifferent between the two options and willing to mix.

3.4.3 When the penalty changes, whose probability changes?

Suppose a shirking worker who is caught is punished more harshly: the payoff of a caught shirker is −2 instead of 0. Which probability changes? The intuitive answer is "the worker works more". The indifference principle says something else.

The penalty changes only the worker's payoff. And the worker's payoffs determine the manager's q. In the new version, the worker's indifference is $2 = -2q + 4(1 - q) = 4 - 6q$, which gives $q^* = 1/3$. Since the manager's payoffs have not changed, the manager's indifference equation stays the same and $p^* = 3/4$ does not change. A harsher penalty does not make the worker work harder; it lets the manager inspect less.

In general, if the payoff of a caught shirker is c, the worker's indifference is $2 = c \cdot q + 4(1 - q)$. This equation can be put into the form $(4 - c)\, q = 2$, and we get

$$q^* = 2/(4 - c), \quad p^* = 3/4$$

At $c = 0$, $q^* = 1/2$; at $c = -2$, $q^* = 1/3$; at $c = -6$, $q^* = 1/5$.

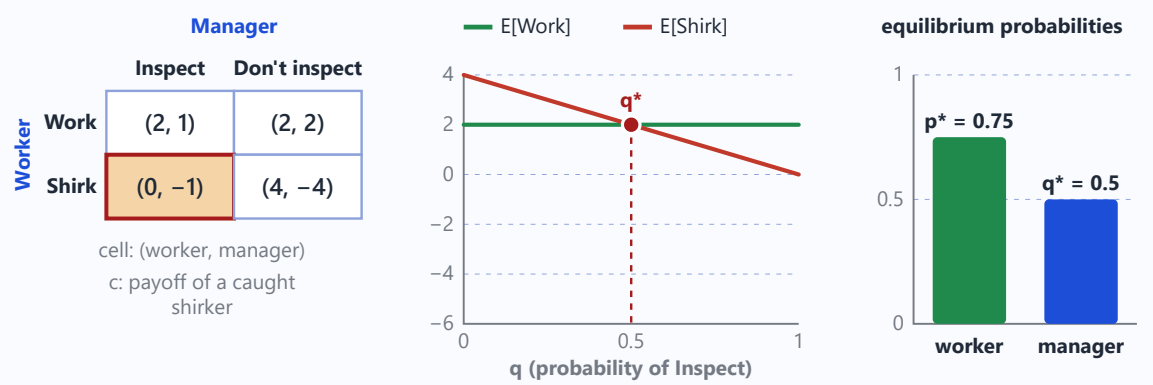


Figure 3.16: The interactive inspection game. The slider changes the payoff c of a caught shirker. The middle chart shows the expected payoffs of the worker's two options, and on the right the equilibrium probabilities: q^* changes, $p^* = 3/4$ stays constant. The default is $c = 0$.

This result is a concrete instance of the principle that "each player's equilibrium probability is determined by the opponent's payoffs". The result also has a policy meaning. When the penalty rises, the probability of inspection falls and the cost of inspection is saved, but the rate of rule-breaking does not change. To change that rate, the inspector's (the manager's) payoffs would have to change.

3.4.4 Hide and seek: repeating the method

Let us apply the indifference method once more, writing out each step. Ayşe hides (row) and Bora seeks (column). The options are Garden and House. Each cell shows the enjoyment points (Ayşe, Bora). If Bora finds Ayşe in the garden, he gets 3 points; in the house, 1 point. If Ayşe is not found in the garden she gets 2 points; if she is not found in the house, 1 point. A found Ayşe gets 0, and a Bora who fails to find her gets 0.

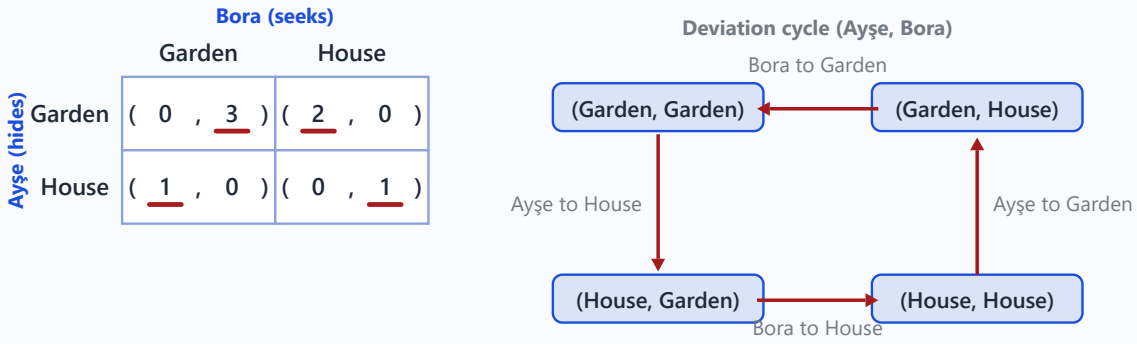


Figure 3.17: The hide-and-seek game. On the left, the payoff matrix with underlined best responses; no cell has two underlines. On the right, the deviation cycle formed by the best responses.

Example 3.11

1. **Is there a pure equilibrium?** After underlining, no cell has two underlines. At (Garden, Garden) Ayşe switches to House; at (House, Garden) Bora switches to House; at (House, House) Ayşe switches to Garden; at (Garden, House) Bora switches to Garden. There is no pure equilibrium.
2. **Variables.** Ayşe chooses Garden with probability p and Bora chooses Garden with probability q. If Ayşe mixes, her two options must give her equal payoffs; this equality yields q.
3. **Ayşe in the Garden.** With probability q Bora looks in the garden and Ayşe gets 0; with probability $1 - q$ he looks in the house and Ayşe gets 2. $E[Garden] = 0 \cdot q + 2(1 - q) = 2 - 2q$.
4. **Ayşe in the House.** If Bora looks in the garden (q), Ayşe gets 1; if he looks in the house, 0. $E[House] = 1 \cdot q + 0 \cdot (1 - q) = q$.
5. **Equate and solve.** $2 - 2q = q \Rightarrow 3q = 2 \Rightarrow q^* = 2/3$. Bora must look in the garden with probability 2/3 so that Ayşe stays indifferent.
6. **The same for Bora.** If Bora looks in the Garden, he gets 3 when Ayşe is there (p) and 0 when she is not: $E[Garden] = 3p$. If he looks in the House: $E[House] = 0 \cdot p + 1 \cdot (1 - p) = 1 - p$.
7. **Equate.** $3p = 1 - p \Rightarrow 4p = 1 \Rightarrow p^* = 1/4$. p^* came out of Bora's numbers, not Ayşe's.
8. **Check.** $0 < 1/4 < 1$ and $0 < 2/3 < 1$ ✓. Expected payoffs: Ayşe $2 - 2 \cdot (2/3) = 2/3$, Bora $3 \cdot (1/4) = 3/4$.

At first sight the result looks contradictory: Ayşe, who likes the garden more (2 points if she is not found there), hides there with probability only 25%. The reason is that Bora also likes finding her in the garden more (3 points). Bora's wish to look in the garden drives Ayşe away from it. Ayşe's liking for the garden, in turn, determines Bora's behavior: Bora looks in the garden with probability 2/3.

3.4.5 Code trace: mixed_2x2()

The indifference equations can be solved in general for every 2 × 2 game. Let the row player's payoffs be A and the column player's B (index 0 is the first option, 1 the second). The row player's indifference is the equality

$$A_{00}\,q + A_{01}(1-q) = A_{10}\,q + A_{11}(1-q)$$

Collecting the terms in q on one side gives $q\,(A_{00} - A_{01} - A_{10} + A_{11}) = A_{11} - A_{01}$. Calling the number in parentheses d_a, we get $q = (A_{11} - A_{01}) / d_a$. Doing the same for the column player's indifference $B_{00}\,p + B_{10}(1-p) = B_{01}\,p + B_{11}(1-p)$ gives $p = (B_{11} - B_{10}) / d_b$, with $d_b = B_{00} - B_{01} - B_{10} + B_{11}$. The function below evaluates these formulas with fractions (`Fraction`), so results such as 2/5 come out without decimal rounding error.

```
from fractions import Fraction as F
def mixed_2x2(A, B):
    da = A[0][0] - A[0][1] - A[1][0] + A[1][1]
    if da == 0: return None     # degenerate
    q = F(A[1][1] - A[0][1], da)  # row indifferent
    db = B[0][0] - B[0][1] - B[1][0] + B[1][1]
    if db == 0: return None
    p = F(B[1][1] - B[1][0], db)  # column indifferent
    if not (0 < p < 1 and 0 < q < 1):
        return None               # no interior eq.
    u1 = A[0][0] * q + A[0][1] * (1 - q)
    return p, q, u1
```

The function returns `None` in three cases: if $d_a = 0$, q drops out of the row player's indifference equation (the difference between the two options does not depend on q; either no q makes the row player indifferent or every q does, so `None` does not mean "no equilibrium"); if $d_b = 0$, the same happens for the column player; and if one of the probabilities is not between 0 and 1, there is no interior mixed equilibrium. In the last line, `u1` is the row player's expected payoff when playing the first row. By the indifference principle the second row gives the same value, so this number is the row player's equilibrium payoff.

Let us run the function on the Battle of the Sexes. Let 0 = Opera and 1 = Football; `A = [[3, 0], [0, 2]]` holds the row player's payoffs and `B = [[2, 0], [0, 3]]` the column player's. The row player likes the Opera more and the column player likes Football more; both want to avoid ending up apart (0).

Line	Computation	Value	Meaning
`da`	3 − 0 − 0 + 2	5	≠ 0, the equation exists
`q`	(2 − 0) / 5	2/5	3q = 2(1 − q): the column player plays Opera with probability 2/5
`db`	2 − 0 − 0 + 3	5	≠ 0

Line	Computation	Value	Meaning
`p`	(3 − 0) / 5	3/5	2p = 3(1 − p): the row player plays Opera with probability 3/5
check	0 < 3/5 < 1, 0 < 2/5 < 1	true	an interior mixed equilibrium exists
`u1`	3 · 2/5 + 0 · 3/5	6/5	the row player's expected payoff
return		(3/5, 2/5, 6/5)	

Table 3.14: Trace of `mixed_2x2(A, B)`, Battle of the Sexes.

The column player's expected payoff is also $u_2 = 2 \cdot (3/5) = 6/5$. This number is striking: at the two pure equilibria (Opera, Opera) and (Football, Football) the players get 2 or 3. At the mixed equilibrium both get only 6/5, because they often end up apart. Failing to meet happens when the row player goes to the Opera and the column player to Football, or the reverse. Since the column player's probability of Football is $1 - 2/5 = 3/5$ and the row player's is $1 - 3/5 = 2/5$, the probability of failing to meet is $(3/5)(3/5) + (2/5)(2/5) = 9/25 + 4/25 = 13/25 = 52\%$.

The `None` branches of the function are best understood through examples. In the **Free Money** game only the (Heads, Heads) cell is (1, 1) and the other cells are 0. For `A = [[1, 0], [0, 0]]` we get $d_a = 1$ and $q = (0 - 0)/1 = 0$; since q is not strictly between 0 and 1, the function returns `None`. In the **Prisoner's Dilemma**, for `A = [[-1, -3], [0, -2]]` we get $d_a = -1 + 3 - 0 - 2 = 0$, and the first check returns `None`: because Betray gives exactly 1 more than Silent in every case, the row player is indifferent at no q.

3.4.6 Mixed equilibria of six classic games

The table below compares the equilibria of six classic 2 × 2 games. p^* and q^* are the probabilities of playing the first strategy (Heads, Swerve, Opera, Stag, Party, Heads).

Game	Pure NE	Mixed NE (p*, q*)	(u_1, u_2)	What happens at the mixed equilibrium?
Matching Pennies	0	1/2, 1/2	(0, 0)	Each of the four outcomes has probability 25%; this is the only equilibrium, and nobody can be exploited.
Chicken (intersection)	2	9/10, 9/10	(−1/10, −1/10)	Collision $(1/10)^2 = 1\%$; if a crash cost −k, the probability of swerving would be (k − 1)/k.
Battle of the Sexes	2	3/5, 2/5	(6/5, 6/5)	Failing to meet: 0.36 + 0.16 = 52%.
Stag Hunt (stag 4, hare 3)	2	3/4, 3/4	(3, 3)	Ending up apart 3/8 = 37.5%; stag together 9/16 ≈ 56%.
Pure Coordination	2	1/3, 1/3	(10/3, 10/3)	Different decisions 4/9 ≈ 44%; the more attractive the party, the less often it is chosen.
Free Money	2	none (only the boundary: 0, 0)	—	If q > 0, Heads is strictly better; indifference only at q = 0.

Table 3.15: Mixed equilibria of six classic games. Chicken: (Swerve, Swerve) = (0, 0), (Swerve, Straight) = (−1, 1), (Straight, Straight) = (−10, −10). Pure Coordination: (Party, Party) = (10, 10), (Home, Home) = (5, 5), other cells 0. Free Money: only (Heads, Heads) = (1, 1), other cells 0.

Let us derive two of the numbers in the table in detail.

Example 3.12

Chicken (intersection). Two drivers arrive at an intersection without traffic lights at the same time. Each chooses Swerve (give way) or Straight. Suppose the other driver swerves with probability x.

- $E[\text{Swerve}] = 0 \cdot x + (-1)(1 - x) = x - 1$.
- $E[\text{Straight}] = 1 \cdot x + (-10)(1 - x) = 11x - 10$.
- Indifference: $x - 1 = 11x - 10 \Rightarrow 9 = 10x \Rightarrow x = 9/10$.

Because the game is symmetric, both drivers swerve with probability 9/10. The expected payoff is $9/10 - 1 = -1/10$. The probability that both drivers go straight (a collision) is $(1/10)^2 = 1/100$. If a crash cost $-k$, indifference would give $x - 1 = x - k(1 - x)$, that is, $x = (k - 1)/k$. As the cost grows, drivers swerve more often, but the collision probability $(1/k)^2$ never becomes zero.

Pure Coordination. If the other side chooses Party with probability q, the expected payoff of Party is $10q$ and that of Home is $5(1 - q)$. $10q = 5 - 5q \Rightarrow q = 1/3$. Although Party is more attractive, it is chosen with probability 1/3. The reason is the indifference principle: the larger the payoff of Party, the more *rarely* Party must be chosen in order to leave the other side indifferent. The probability of different decisions is $2 \cdot (1/3)(2/3) = 4/9$.

The last row of the table is a warning. In Free Money, as long as the column player chooses Heads with probability $q > 0$, Heads (payoff q) is strictly better for the row player than Tails (payoff 0). Indifference occurs only at $q = 0$, which is not a "mixture" but the pure (Tails, Tails) equilibrium. In games like this, with ties in the payoffs, there may be no interior mixed equilibrium.

3.4.7 Common mistakes with mixed equilibria

Common mistake

- **Finding p from one's own numbers:** The row player's p makes the column player indifferent; so it is computed from the column player's payoffs.
- **Forgetting (1 − q):** $E = 0 \cdot q + 2 \cdot (1 - q) = 2 - 2q$, not "2q". Each term is multiplied by its own probability.
- **Not checking the result:** If p or q does not come out between 0 and 1, there is no interior mixed equilibrium (usually a strategy is dominated).
- **Skipping pure equilibria:** A mixed computation does not replace the pure equilibria; underline first, then look for the mixed equilibrium.

- **"If indifferent, any p will do":** A player is indifferent among their own options, but in equilibrium they play the one p that makes the opponent indifferent. If they played another p, the opponent would move to a pure best response and the equilibrium would break.
- **Setting up the equality wrongly:** What is equated is the expected payoffs of the same player's two options; the payoffs of the two players are not equated.

3.5 Existence and equilibrium selection

Some of the games examined so far had a single equilibrium and some had several; in games with no pure equilibrium, a mixed equilibrium was found. Two questions arise. Does every game have an equilibrium? And if there are several, which one is played?

3.5.1 Nash's existence theorem

Theorem 3.4 (Nash's existence theorem, 1950)

Every game with a finite number of players, each with a finite strategy set, has at least one Nash equilibrium (in mixed strategies).

The full proof of the theorem is beyond the scope of this course, but the idea can be understood. It rests on the notion of a **fixed point**. A fixed point of a function is a point the function leaves unchanged: $f(x) = x$. In its simplest form: every continuous function (one whose graph can be drawn without lifting the pen) that maps the interval [0, 1] into [0, 1] has a fixed point. The graph starts on or above the diagonal at $x = 0$ and ends on or below it at $x = 1$; since it is drawn without lifting the pen, it must cross the diagonal somewhere. That crossing point is the fixed point. This is the one-dimensional case of Brouwer's fixed-point theorem.

In Nash's proof the "point" is a mixed-strategy profile. In place of the function there is the best-response correspondence: it assigns to each profile the profiles in which everyone best-responds to the others. A fixed point of the correspondence is a profile in which everyone is already best-responding to the others; by Theorem 3.1, this is a Nash equilibrium. Kakutani's fixed-point theorem (the generalization of Brouwer's to correspondences that return sets instead of single values) guarantees that this correspondence has a fixed point.

This idea also shows why mixed strategies are needed here. Fixed-point theorems require "continuity": probabilities can change continuously (from 0.30 to 0.31), whereas pure strategies change by jumps (from Heads to Tails). Restricted to pure strategies, there are gaps, and the equilibrium can disappear. Indeed, Matching Pennies has no pure equilibrium. So the theorem's guarantee is for mixed strategies.

The finiteness condition is also necessary. If a strategy set is infinite, there may be no equilibrium. For example, let two players simultaneously name a natural number, and let the one who names the larger number win. There is always a larger number than any given one; whichever profile is chosen, the loser can win by naming a larger number. No profile is an equilibrium. In the Cournot game the strategy set was infinite, yet an equilibrium existed, because there the payoffs were smooth and the best responses stayed in a bounded interval. For infinite sets, existence depends on additional conditions.

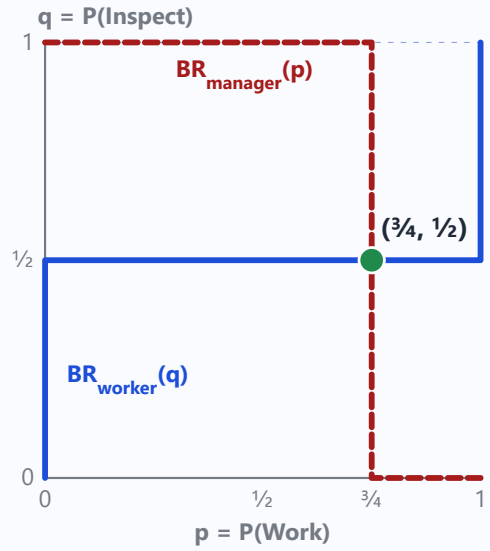


Figure 3.18: Best-response curves in the inspection game. The solid blue curve is the worker's best response $BR_{worker}(q)$, the dashed red curve is the manager's best response $BR_{manager}(p)$. The single intersection point (3/4, 1/2) is the equilibrium.

Figure 3.18 makes this idea concrete in the inspection game. The horizontal axis is the worker's probability p of Working, and the vertical axis is the manager's probability q of Inspecting. The worker's best response is as follows: Shirk ($p = 0$) when $q < 1/2$, Work ($p = 1$) when $q > 1/2$, and at $q = 1/2$ the worker is indifferent and every p is a best response. So the blue curve has a "step" shape and contains a horizontal segment at $q = 1/2$. The manager's best response is a similar step: Inspect ($q = 1$) when $p < 3/4$, Don't inspect ($q = 0$) when $p > 3/4$. The two steps cross at a single point. At the crossing point both players best-respond to each other; this point is the equilibrium. That the curves must cross is the fixed-point theorem showing up in this simple game.

3.5.2 Three equilibria in channel sharing

Two wireless devices want to send packets on the same channel. If both send at the same time, the packets collide and each gets −2. If only one sends, the sender gets 3 and the one that waits gets 1 (the waiting device also benefits from the channel working properly). If both wait, the channel goes unused and both get 0. This structure is the same as the Chicken game: each player wants to do the opposite of the other.

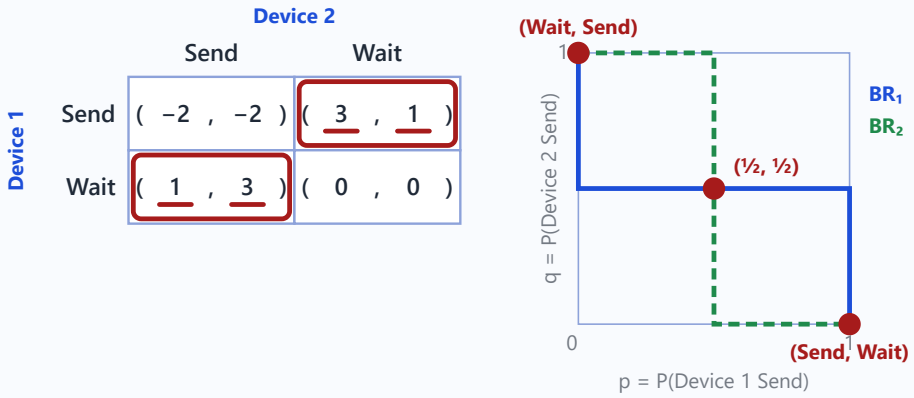


Figure 3.19: Channel sharing. On the left, the payoff matrix; the cells with two underlines, (Send, Wait) and (Wait, Send), are the pure equilibria. On the right, the best-response curves: BR_1 (blue) and BR_2 (green, dashed) intersect at three points.

Example 3.13

1. **Device 1's best responses** (first numbers). If the other device Sends: Send −2, Wait 1; since −2 < 1, the best response is Wait. If the other Waits: Send 3, Wait 0; the best response is Send.
2. **The same for device 2** (second numbers). The cells with two underlines are (Send, Wait) = (3, 1) and (Wait, Send) = (1, 3): two pure equilibria.
3. **Mixed equilibrium.** Let q = P(device 2 Sends). Device 1's expected payoff from Send is $-2q + 3(1 - q) = 3 - 5q$.
4. Its expected payoff from Wait: 1 if the other sends, 0 if it waits, that is, $1 \cdot q + 0 \cdot (1 - q) = q$.
5. **Indifference:** $3 - 5q = q \Rightarrow 6q = 3 \Rightarrow q^* = 1/2$. Because the game is symmetric, device 2's indifference also gives $p^* = 1/2$.
6. **Results.** At the mixed equilibrium each of the four cells has probability 1/4. The expected payoff is $¼ \cdot (-2) + ¼ \cdot 3 + ¼ \cdot 1 + ¼ \cdot 0 = ½$.

The best-response curves (Figure 3.19) intersect at three points: the two corners are the pure equilibria, and the middle point is the mixed equilibrium. Because it is symmetric, the mixed equilibrium is "fair", but it is inefficient. In a quarter of the time the packets collide, in a quarter the channel sits idle, and each device earns only ½ on average. At the pure equilibria each device gets 3 or 1. Random backoff in network protocols is the practical counterpart of this randomization: colliding devices wait a random time and try again, so the question "who sends first" is settled without coordination.

3.5.3 How many equilibria? The odd-number rule

There is an interesting pattern in the numbers of equilibria of the games in this chapter.

Game	Pure equilibria	Mixed equilibria	Total
Prisoner's Dilemma	1	0	1
Hide and Seek, Inspection, Matching Pennies	0	1	1
Channel sharing, Chicken, Battle of the Sexes, Stag Hunt, Pure Coordination	2	1	3
Language choice (3 × 3)	2	1 (Ayşe plays Python with probability 3/4, Bora with probability 1/5)	3
The trap of weak domination (with ties)	3	infinitely many	—
Free Money (with ties)	2	0	2 (even)

Table 3.16: Numbers of equilibria of the games in this chapter.

Theorem 3.5 (Oddness, Wilson 1971)

In almost all finite games, that is, games whose payoffs are in general position (**nondegenerate** games), the number of Nash equilibria is finite and odd (1, 3, 5, ...). Special ties among payoffs are the best-known way this condition fails.

The proof of the theorem is advanced (Wilson, 1971), but it gives a practical rule: if two pure equilibria have been found, there is probably one more, mixed, equilibrium, and it should be searched for with the indifference method. The language-choice game is a nice example. Besides the two pure equilibria, there is a mixed equilibrium in which the C options are never played. Ayşe plays Python with probability 3/4 and Bora plays Python with probability 1/5. Against this mixture, Ayşe's Python and Java rows give 8/5 and her C row gives only 1/5; Bora's Python and Java columns give 7/4 and his C column gives 1/2. Switching to C gains nobody anything, so the mixture found in the 2 × 2 game is also an equilibrium of the 3 × 3 game.

In games with ties the rule can fail. In the trap of weak domination, the column player gets 1 in every cell and so accepts any mixture; for example, while the row player plays Bottom, every mixture of the column player is an equilibrium. There are infinitely many equilibria. In Free Money the number of equilibria is 2, which is even. These games are "degenerate": the ties among payoffs violate the assumption of the rule. If the payoffs are changed slightly (for example, one tie is made 1.01 instead of 1), the odd-number rule returns.

3.5.4 Multiple equilibria: Stag Hunt

The **Stag Hunt**, based on a story told by the philosopher Jean-Jacques Rousseau, is the best-known example of multiple equilibria. Two hunters decide simultaneously whether to hunt Stag or Hare. The stag is caught only if both hunt it together, and each gets 4. A hare can be caught alone, and guarantees 3 whatever the partner does. A hunter who goes for the stag alone comes home empty-handed (0).

	Stag	Hare
Stag	**(4, 4)**	(0, 3)
Hare	(3, 0)	**(3, 3)**

Table 3.17: Stag Hunt. Hunter 1 is the row player and Hunter 2 the column player; each cell shows (Hunter 1, Hunter 2).

The best responses coincide in two cells: (Stag, Stag) and (Hare, Hare). If the partner goes for the stag, going for the stag is best (4 > 3); if the partner goes for the hare, going for the hare is best (3 > 0). The third equilibrium is mixed: if the other side chooses Stag with probability q, the expected payoff of Stag is $4q$ and that of Hare is 3 in every case. $4q = 3 \Rightarrow q = 3/4$. Stag is a best response only if one believes the partner will come with probability at least 75%. Which equilibrium will be played?

Two criteria give different answers.

Pareto dominance: One equilibrium Pareto dominates another if it gives everyone at least as much and at least one player more. (Stag, Stag) gives both hunters more (4 > 3). The efficiency criterion selects (Stag, Stag).

Risk dominance: This criterion, proposed by Harsanyi and Selten (1988), asks which equilibrium is "safer" when one is unsure what the partner will do. At each equilibrium, the loss from a unilateral deviation is computed for every player and these losses are multiplied. The equilibrium with the larger product is risk dominant.

- At (S, S), a hunter who deviates to Hare gets 3 instead of 4: the loss is $4 - 3 = 1$. For the two hunters the product is $1 \cdot 1 = 1$.
- At (H, H), a hunter who deviates to Stag gets 0 instead of 3: the loss is $3 - 0 = 3$. The product is $3 \cdot 3 = 9$.

Since $9 > 1$, (Hare, Hare) is risk dominant. The intuition is this: at (H, H) the cost of being wrong is low (the hare is guaranteed anyway), whereas at (S, S) misplaced trust in the partner is costly (coming home empty-handed). In 2 × 2 symmetric games, the risk-dominant equilibrium is the one that is a best response to a 50–50 belief about what the partner will do: if the partner comes for the stag with probability ½, Stag gives $4 \cdot \tfrac{1}{2} = 2$ and Hare gives 3.

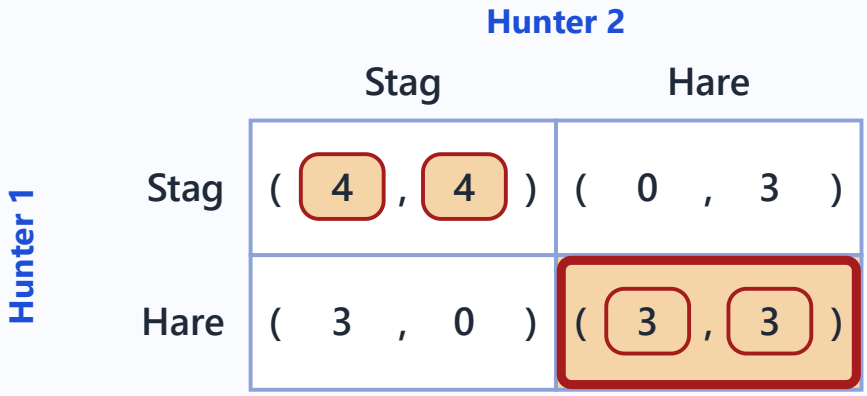


Figure 3.20: Risk dominance in Stag Hunt. (Stag, Stag) is Pareto dominant and (Hare, Hare) is risk dominant (red frame): the product of deviation losses is 9, larger than that of (Stag, Stag), which is 1.

3.5.5 Criteria for equilibrium selection

The theory alone does not say which of several equilibria will be played. This question is called the problem of **equilibrium selection**. The criteria that stand out in the literature are the following:

Criterion	What it says	Example
Pareto dominance	Choose the equilibrium that makes everyone better off. Requires trust.	(S, S) in Stag Hunt.
Risk dominance	If you are not sure what the other side will do, choose the one with the lower risk of loss.	(H, H) in Stag Hunt.
Focal point (Schelling, 1960)	Culture, habit, or salience makes one equilibrium stand out.	"If we are to meet in Istanbul: Taksim at noon."
Correlated equilibrium (Aumann, 1974)	A common signal steers the players to one of the equilibria.	A traffic light resolves Chicken at an intersection.
Learning and evolution	In repeated play, players learn by trial; the equilibrium reached depends on initial behavior.	If everyone starts out hunting hare, the stag equilibrium may never be reached.

Table 3.18: Criteria for equilibrium selection.

The **focal point** is especially interesting. Two friends are to meet in Istanbul but have not been able to discuss the place and time. Every common place and time is an equilibrium; the payoffs are equal, so neither the Pareto nor the risk criterion helps. Even so, many people answer "Taksim at noon". What determines the choice is not the numbers but shared culture and salience. Thomas Schelling put forward this observation in 1960.

Equilibrium selection also raises a **problem of fairness**. In the Battle of the Sexes, one pure equilibrium favors one player (Opera: 3 to 2) and the other favors the other (Football: 2 to 3). The symmetric mixed equilibrium, which looks "fair", gives each only 6/5; for both players it is worse

than either pure equilibrium. Fairness and efficiency conflict here. In real life such conflicts are resolved by taking turns (opera one week, football the next) or by an outside signal such as a coin flip. The idea of an outside signal is the core of the correlated equilibrium in Section 3.5.7.

3.5.6 An opponent who makes mistakes

The equilibrium concept assumes that everyone correctly knows the others' behavior. Let us relax this assumption a little: suppose the opponent, with a small probability ε, makes a move they did not intend (a slip of the hand, a wrong key). Does the best response change?

Prisoner's Dilemma: no change. Let the opponent stay Silent with probability p. The expected payoff of Sam staying Silent is $-1 \cdot p - 3 \cdot (1 - p) = 2p - 3$, and that of Betray is $0 \cdot p - 2 \cdot (1 - p) = 2p - 2$. The difference is $(2p - 2) - (2p - 3) = 1$ for every p. Betray is exactly 1 year better for every p; the decision does not change whatever the opponent's error rate.

Stag Hunt: depends on the error rate. Let the opponent slip to Hare with probability q. The expected payoff of Stag is $4 \cdot (1 - q)$, and that of Hare is 3 in every case. Stag is a best response only when $4(1 - q) \geq 3$, that is, $q \leq 1/4$. If the partner's probability of slipping to hare exceeds 25%, the safe choice, the risk-dominant Hare, is the better one.

The general lesson is this: a dominant-strategy equilibrium is robust to mistakes; coordination equilibria are not. A dominant strategy does not depend at all on what the opponent does, so the opponent's mistakes do not affect it. A coordination equilibrium, on the other hand, rests on trust that the other side will play correctly; as trust declines, the equilibrium is shaken.

3.5.7 Correlated equilibrium

Two drivers arrive at an intersection without traffic lights at the same time. Each either gives way (Swerve) or goes through (Straight). This is a Chicken game: (Swerve, Swerve) = (0, 0), (Swerve, Straight) = (−1, 1), (Straight, Swerve) = (1, −1), (Straight, Straight) = (−10, −10). Each of the two pure equilibria favors one driver; without coordination, in the symmetric mixed equilibrium each driver swerves with probability 9/10, a collision occurs with probability 1%, and the expected payoff is −1/10 (the Chicken example).

Now put a traffic light at the intersection. The light works like a mediator: it draws a cell from a publicly announced probability distribution μ and whispers to each driver *only their own recommendation*. Here μ gives probability 1/3 each to the cells (Swerve, Swerve), (Swerve, Straight), and (Straight, Swerve), and probability 0 to (Straight, Straight).

Definition 3.8

A probability distribution μ over profiles is a **correlated equilibrium** (CE) if it satisfies the following condition: for every player and every strategy that may be recommended to them, when the player knows only their own recommendation, the expected payoff from obeying the recommendation is no less than the expected payoff from deviating from it. These conditions are called **obedience constraints**.

Checking the obedience constraints requires **conditional probability**. The probability of an event given that another event has occurred is $P(A \mid B) = P(A \text{ and } B) / P(B)$. **Bayes' rule** rests on this definition; when a driver sees their own recommendation, they update their belief about the opponent's recommendation in this way.

Example 3.14

1. **If the recommendation is "Swerve".** The cells in which driver 1 is told Swerve are (Swerve, Swerve) and (Swerve, Straight); each has probability 1/3, for a total of 2/3. The conditional probabilities are $(1/3)/(2/3) = 1/2$ each: the opponent has received the recommendation Swerve with probability ½ and Straight with probability ½.
2. If the driver obeys (Swerve): $½ \cdot 0 + ½ \cdot (-1) = -½$. If the driver deviates (Straight): $½ \cdot 1 + ½ \cdot (-10) = -4.5$. $-½ > -4.5$; obeying is better.
3. **If the recommendation is "Straight".** The only cell in which driver 1 is told Straight is (Straight, Swerve); the opponent has certainly been told "Swerve". Obeying gives 1, deviating (Swerve) gives 0. Obeying is better.
4. Because the game is symmetric, driver 2's constraints are satisfied in the same way. Deviating from a recommendation is never profitable;μ is a correlated equilibrium.
5. **Payoff.** Each driver's expected payoff is $⅓ \cdot 0 + ⅓ \cdot (-1) + ⅓ \cdot 1 = 0$. This is better than the −1/10 of the mixed equilibrium, and there is never a collision.

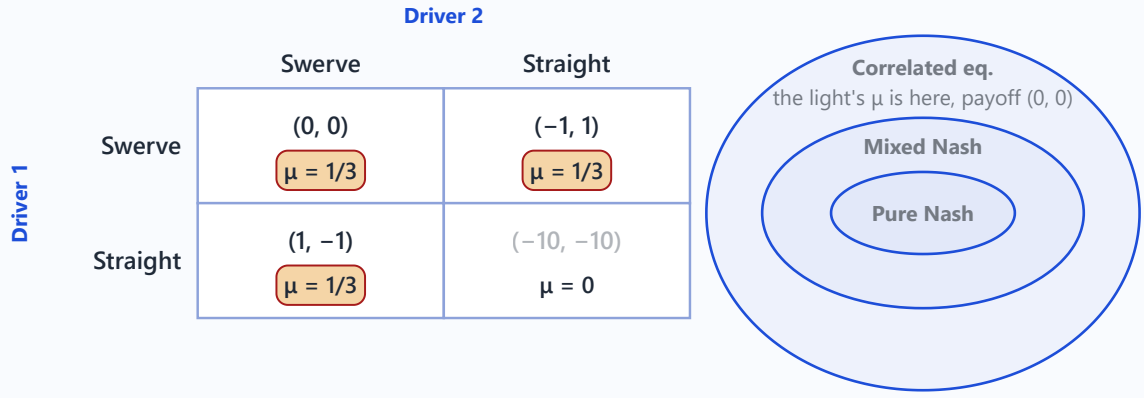


Figure 3.21: A traffic light at the intersection. On the left, the light's distribution μ: $1/3$ on three cells, 0 on the collision cell. On the right, the nesting of the sets: every pure Nash equilibrium is a mixed Nash equilibrium, and every Nash equilibrium is a correlated equilibrium.

The light's power comes from the fact that drivers know nothing beyond their own recommendations. A driver who is told to Swerve knows that the opponent may also have been told to Swerve; this uncertainty makes deviation costly. The mediator enforces nothing; obeying is in everyone's own interest.

The correlated equilibrium matters for two reasons. First, its scope is broad: every Nash equilibrium is a correlated equilibrium (if the mediator recommends exactly the players' independent mixtures, the obedience constraints reduce to the Nash condition), but the converse is false. The light in this example corresponds to no Nash equilibrium. Second, it is easy to compute: all the obedience constraints are linear in the probabilities of μ. A correlated equilibrium can therefore be found in polynomial time by **linear programming** (LP); no such method is known for the Nash equilibrium (Section 3.6.6). Tuning the light's timing to traffic density is a separate design decision: which correlated equilibrium is selected is up to whoever sets the rules.

3.6 Minimax and complexity

Two topics are treated together: zero-sum games, in which interests are exactly opposed, and how hard it is for a computer to find an equilibrium in general games. In zero-sum games the equilibrium concept is especially strong and can be computed efficiently.

3.6.1 Zero-sum games and the saddle point

In a zero-sum game the payoffs of the two players in each cell sum to zero: $u_2 = -u_1$. So only the row player's payoff is written in the cell; the column player loses the same amount. The row player wants to make this number larger and the column player wants to make it smaller.

A natural line of reasoning in such a game is pessimistic thinking. The row player says: "Whichever row I choose, my opponent will do the worst to me. So let me look at the worst value of each row and pick the best of those." The column player thinks similarly.

Definition 3.9

In a two-player zero-sum game with row-player payoff matrix A:

- **Maximin:** $max_r\ min_c\ A[r][c]$. The largest payoff the row player can guarantee with pure strategies.
- **Minimax:** $min_c\ max_r\ A[r][c]$. The smallest loss the column player can guarantee with pure strategies.
- If maximin and minimax are equal, the cell that attains this value is called a **saddle point**. A saddle point is a cell that is the smallest in its row and the largest in its column.

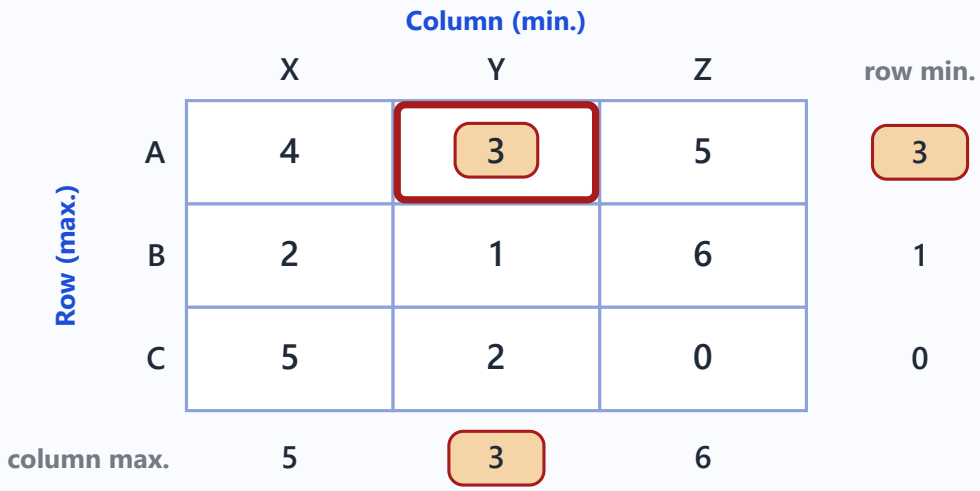


Figure 3.22: Maximin and minimax in a zero-sum game. The right column shows the row minima (3, 1, 0), the bottom row the column maxima (5, 3, 6). Maximin = minimax = 3; the saddle point is (A, Y).

Example 3.15

The matrix: A = (4, 3, 5), B = (2, 1, 6), C = (5, 2, 0); the columns are X, Y, Z.

1. **The row player thinks pessimistically.** In each row, the worst value the opponent can leave them: 3 in A, 1 in B, 0 in C.
2. **Maximin** = max(3, 1, 0) = 3 (row A). By choosing A, the row player guarantees at least 3.
3. **The column player is pessimistic too.** In each column, the highest value the row player can get: 5 in X, 3 in Y, 6 in Z.

4. **Minimax** = min(5, 3, 6) = 3 (column Y). By choosing Y, the column player guarantees losing at most 3.
5. **Result.** Maximin = minimax = 3. The 3 in cell (A, Y) is the smallest in row A and the largest in column Y: a saddle point. This cell is a pure Nash equilibrium, and the **value of the game** is $v = 3$.

It is easy to see that a saddle point is a Nash equilibrium. If the row player switches to another row in column Y, they cannot get more than 3, because 3 is the largest in that column. If the column player switches to another column in row A, they cannot give less than 3, because 3 is the smallest in that row. The name comes from this: a horse's saddle is a peak in one direction (along the column) and a trough in the other (along the row).

Maximin and minimax are not always equal, but maximin can never exceed minimax. The reason is this: let the maximin row be r^* and the minimax column c^*. The minimum of row r^* cannot be larger than $A[r^*][c^*]$, the entry in that row; the maximum of column c^* cannot be smaller than $A[r^*][c^*]$, the entry in that column. Hence maximin $\leq A[r^*][c^*] \leq$ minimax. If equality fails, there is no saddle point and a gap remains between the players' guarantees.

3.6.2 No saddle point: the penalty kick

In a penalty kick, the kicker and the goalkeeper simultaneously choose Left or Right (Left and Right are from the kicker's point of view). Each cell shows the percentage of goals; the kicker wants to make this number larger and the goalkeeper smaller. The numbers are illustrative; the structure in real data is similar (Palacios-Huerta, 2003).

Kicker \ Goalkeeper	Left	Right
Left	60	95
Right	90	70

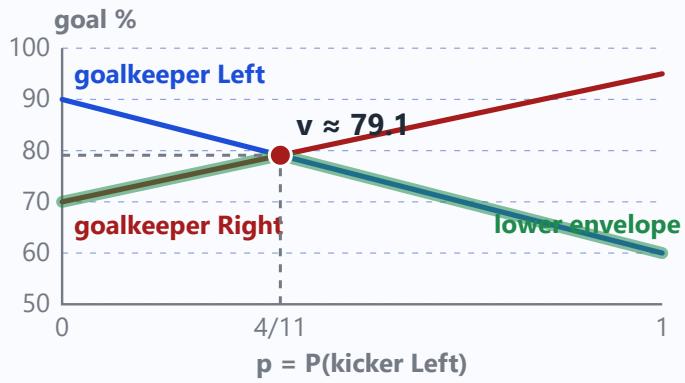


Figure 3.23: The penalty kick. In the matrix, goal percentages: kicker Left–goalkeeper Left 60, kicker Left–goalkeeper Right 95, kicker Right–goalkeeper Left 90, kicker Right–goalkeeper Right 70. In the chart, the two lines as a function of the kicker's probability p = P(Left) and the lower envelope (green); the peak of the envelope is at $p^* = 4/11$, $v \approx 79.1$.

Example 3.16

1. **Pure guarantee.** The row minima are 60 (Left) and 70 (Right): maximin is 70. The column maxima are 90 (goalkeeper Left) and 95 (goalkeeper Right): minimax is 90. $70 < 90$: there is no saddle point.
2. **The kicker mixes.** Let the kicker shoot Left with probability p. If the goalkeeper dives Left, the goal percentage is $60p + 90(1 - p) = 90 - 30p$; if the goalkeeper dives Right, it is $95p + 70(1 - p) = 70 + 25p$.
3. **The kicker's guarantee.** The goalkeeper tries to minimize the goal percentage; so for each p, the kicker's guarantee is the smaller of the two lines. The curve formed by these smaller values is called the **lower envelope**.
4. **The peak of the envelope.** The lower envelope is highest where the two lines cross: $90 - 30p = 70 + 25p \Rightarrow 20 = 55p \Rightarrow p^* = 4/11 \approx 0.36$. The value is $v = 90 - 30 \cdot 4/11 = 870/11 \approx 79.1$.

5. **The goalkeeper's side.** Let the goalkeeper dive Left with probability q. They must leave the kicker indifferent: $60q + 95(1 - q) = 90q + 70(1 - q) \Rightarrow 95 - 35q = 70 + 20q \Rightarrow q^* = 5/11$. The goal percentage is then $95 - 35 \cdot 5/11 = 870/11$.

The goalkeeper's guarantee is also 79.1%. With mixed strategies, max-min = min-max. The kicker guarantees 79.1% on every shot, and the goalkeeper guarantees conceding no more than that.

An interesting detail: the kicker's strong side is Left (95% if the goalkeeper dives the wrong way), yet at equilibrium they shoot Left only with probability 4/11. The goalkeeper dives Left with probability 5/11, that is, less than half the time. By the indifference principle, each player's probability is determined by the opponent's numbers; so the intuition "shoot to the strong side more often" does not hold at equilibrium.

3.6.3 Security games

Security games model situations in which limited resources protect many targets: airports, ports, forest patrols. The defender can protect k of n targets. Which k targets are chosen is a pure strategy, and there are $C(n, k)$ of them. $C(n, k) = n! / (k!\,(n - k)!)$ is the number of ways to choose k items out of n without regard to order. For $n = 20$, $k = 5$ there are 15,504 pure strategies.

The attacker can observe the patrol pattern. An attacker easily gets around a fixed pattern; so the defender randomizes. The ARMOR system (2007), used at Los Angeles International Airport, planned checkpoints and canine patrols with this model (Tambe, 2011).

Example 3.17

One guard, two targets. The airport H has value 4 and the port L has value 2. The guard protects H with probability p and L with probability $1 - p$. An attacker who hits the unprotected target gains its value; one who hits the protected target gets 0.

1. If the attacker hits H, their expected payoff is $4 \cdot (1 - p)$ (4 if H is unprotected).
2. If they hit L it is $2 \cdot p$ (L is unprotected if the guard is at H).
3. The guard leaves the attacker indifferent: $4(1 - p) = 2p \Rightarrow 4 = 6p \Rightarrow p = 2/3$.
4. The attacker's payoff is $4 \cdot (1/3) = 4/3$.

If the guard always protected H, the attacker would hit L and get 2. Randomization lowers the attacker's payoff from 2 to 4/3. The valuable target is protected more often, but no target is left completely unprotected.

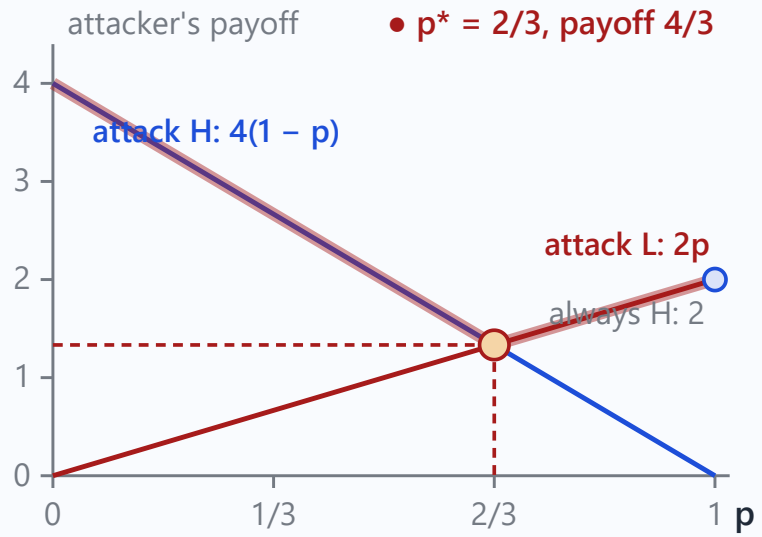


Figure 3.24: A security game. The payoffs of the attacker's two options are plotted against the guard's probability p of protecting H. The attacker picks the larger (upper envelope); the guard picks the lowest point of the envelope, $p^* = 2/3$.

3.6.4 von Neumann's minimax theorem

In the penalty-kick and security examples, the two players' guarantees came out equal under mixed strategies. This is no coincidence. To state the theorem we need the matrix notation for expected payoff. Let the row player's mixed strategy be a probability vector $x = (x_1, \ldots, x_m)$ and the column player's be y. Each entry of the vector Ay is a row's expected payoff against y; and x^TAy is the sum of these row expectations weighted by the probabilities in x, that is, the row player's expected payoff. (x^T denotes x written as a row vector.)

Theorem 3.6 (von Neumann's minimax theorem, 1928)

In every finite two-player zero-sum game,

$$\max_x \min_y x^TAy = \min_y \max_x x^TAy = v$$

holds. v is the value of the game.

The left side is the largest payoff the row player can guarantee with mixed strategies; the right side is the smallest loss the column player can guarantee. With pure strategies there could be a gap between them (70 and 90 in the penalty kick); with mixed strategies the gap always closes. The consequences of the theorem are as follows:

- **Equilibrium = minimax strategies.** (x^*, y^*) is a Nash equilibrium if and only if x^* is a maximin strategy and y^* is a minimax strategy. At every equilibrium the row player's payoff is the same: v.

- **Interchangeability.** If (x, y) and (x', y') are two equilibria, then (x, y') is also an equilibrium. This is because x is a maximin strategy and y' is a minimax strategy, and by the first consequence every such pair is an equilibrium. The players need not coordinate on which equilibrium to play; in zero-sum games there is no equilibrium selection problem.
- **Linear programming.** The row player looks for a mixture x that earns at least v against every column, and for the largest possible v: maximize v; subject to $\Sigma_i\, x_i\, A[i][j] \geq v$ for each column j, $\Sigma_i\, x_i = 1$, and every $x_i \geq 0$. Because all the constraints are linear, this is a linear program, and a computer solves it in polynomial time.
- **Link to learning.** If the players play the game repeatedly and learn by increasing the weight of strategies that did well in the past, their average strategies approach the minimax strategies.

Nash's theorem can be seen as a generalization of the minimax theorem to non-zero-sum n-player games. But the generalization does not come free. In a general game the "value" is not unique (in the Battle of the Sexes the equilibria give different payoffs), interchangeability fails (if one player plays a strategy from one equilibrium and the other from another, the outcome need not be an equilibrium), and an equilibrium cannot be found by linear programming.

3.6.5 Code trace: maximin()

If the row player has only two rows (a $2 \times n$ game), the maximin strategy is easily found by the graphical method. Let the row player play the first row with probability p. Each column j gives a line depending on p: $p \cdot A[0][j] + (1 - p) \cdot A[1][j]$. The column player answers each p with the worst column for the row player; so the row player's guarantee is the lower envelope of these lines: $g(p) = \min_j [p \cdot A[0][j] + (1 - p) \cdot A[1][j]]$. Because the lower envelope is piecewise linear, its highest point is found either at the endpoints ($p = 0$, $p = 1$) or at an intersection of two lines. The function collects these candidates and picks the best.

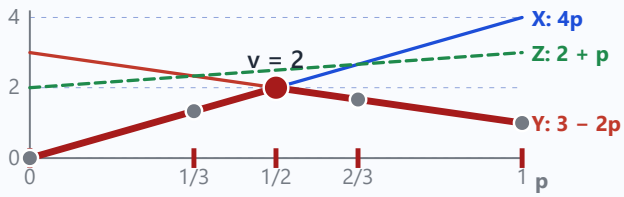
```
from fractions import Fraction as F
def maximin(A):                   # A: 2 x n
    n, cand = len(A[0]), [F(0), F(1)]
    for j in range(n):
        for k in range(j + 1, n):
            d = A[0][j] - A[1][j] - A[0][k] + A[1][k]
            if d != 0:
                cand.append(F(A[1][k] - A[1][j], d))
    g = lambda p: min(p*a+(1-p)*b for a, b in zip(*A))
    p = max((c for c in cand if 0 <= c <= 1), key=g)
    return p, g(p)
```

The intersection of two lines is found as follows. Equate the lines of columns j and k: $p\, A[0][j] + (1 - p)\, A[1][j] = p\, A[0][k] + (1 - p)\, A[1][k]$. Rearranging gives $p \cdot d = A[1][k] - A[1][j]$, where $d = A[0][j] - A[1][j] - A[0][k] + A[1][k]$. If $d = 0$ the lines are parallel and do not intersect. The expression `zip(*A)` yields the columns of the matrix as pairs (a, b): a is the entry in the first row and b the entry in the second row.

Let the example matrix be `A = [[4, 1, 3], [0, 3, 2]]`; the columns are X, Y, Z. The lines are X: $4p$, Y: $p + 3(1 - p) = 3 - 2p$, Z: $3p + 2(1 - p) = 2 + p$. The pure maximin is 1 and the minimax is 3; there is no saddle point.

Step	Computation	Result
Start	`n = 3`, endpoints	`cand = [0, 1]`
X and Y	d = 4 − 0 − 1 + 3 = 6; p = (3 − 0)/6	1/2 is added
X and Z	d = 4 − 0 − 3 + 2 = 3; p = (2 − 0)/3	2/3 is added
Y and Z	d = 1 − 3 − 3 + 2 = −3; p = (2 − 3)/(−3)	1/3 is added
g(0)	min(0, 3, 2)	0
g(1)	min(4, 1, 3)	1
g(1/2)	min(2, 2, 5/2)	2 (best)
g(2/3)	min(8/3, 5/3, 8/3)	5/3
g(1/3)	min(4/3, 7/3, 7/3)	4/3
return	`max(..., key=g)`	(1/2, 2)

Table 3.19: Trace of `maximin(A)`.



candidate p	0	1	1/2	2/3	1/3
g(p)	0	1	2	5/3	4/3

Figure 3.25: The final state of the `maximin()` trace. The thick red line is the lower envelope $g(p)$; the candidate points are marked. The largest guarantee is $v = 2$ at $p = 1/2$. The line of Z (green, dashed) stays above the envelope.

The largest guarantee is $v = 2$ at $p = 1/2$. The line of column Z gives 2.5 at this point and stays above the envelope; the column player never plays Z. The minimax side can be checked too: if the column player plays X with probability 1/3 and Y with probability 2/3, the row player's first row gives $4/3 + 2/3 = 2$ and the second row gives $0 + 2 = 2$. The row player can get at most 2. Max-min = min-max = 2.

3.6.6 How hard is it to compute an equilibrium?

In computer science, a problem counts as "efficiently solvable" if, as the input grows, the running time grows like a polynomial in the input size (for example n^2 or n^3). If the time grows exponentially, like 2^n, even a small increase in size makes the computation infeasible. For the Nash equilibrium the picture is as follows:

Problem	Complexity	Method / source
Pure Nash equilibrium (matrix given)	Polynomial	Best-response marking, O(mn)

Problem	Complexity	Method / source
Two-player zero-sum, mixed	Polynomial	Linear programming (minimax)
Two-player general game, mixed	PPAD-complete	Chen–Deng (2006)
Three or more players, exact equilibrium	FIXP-complete	Etessami–Yannakakis (2007)
Three or more players, ε-approximate equilibrium (ε small enough)	PPAD-complete	Daskalakis–Goldberg–Papadimitriou (2006)
Highest-welfare equilibrium / is it the only one?	NP-hard	Gilboa–Zemel (1989); Conitzer–Sandholm (2008)

Table 3.20: The complexity of equilibrium problems.

The first two rows of the table are the methods we have seen in this chapter, and they are efficient. In practice two methods are used to find mixed equilibria in general games. **Support enumeration** guesses a possible support set for each player and solves the indifference equations; this is the method of Section 3.4.5 tried over all pairs of supports. The **Lemke–Howson algorithm** follows a path from a starting point to an equilibrium. Both methods often work well in practice, but in the worst case they take exponential time: the number of supports grows exponentially with the number of strategies.

In games with three or more players there is one more problem: the equilibrium probabilities may be irrational numbers (for example, numbers involving $\sqrt{2}$) and cannot be written exactly on a computer. In two-player games the indifference equations are linear; if the payoffs are rational, there is always at least one equilibrium with rational probabilities (writable as fractions); with three players, products of probabilities appear in the equations and the roots may turn out irrational. So in games with many players one looks for an **ε-approximate equilibrium**, that is, a profile in which no deviation gains more than ε.

The last row of the table is the difficulty of searching for a "particular" equilibrium instead of "any" equilibrium: finding the equilibrium with the highest social welfare, or deciding whether the game has only one equilibrium, is NP-hard. For further reading: for a fixed ε (for example 0.1) there is a quasi-polynomial-time algorithm for two-player games (Lipton, Markakis and Mehta, 2003); for games with many players, the problem remains PPAD-hard even for constant ε (Rubinstein, 2015).

3.6.7 PPAD

When we speak of hard problems, NP-completeness usually comes to mind. NP-complete problems are questions of the form "does a solution exist?" (for example, "is there a path in this graph that visits every vertex exactly once?"). For the Nash equilibrium this question is trivial: Nash's theorem says an equilibrium always exists, so the answer is always "yes". NP-completeness is therefore not a suitable measure for problems like this, whose solution is guaranteed.

The class **PPAD** (Papadimitriou, 1994) is defined for exactly this situation. PPAD is the class of problems for which a solution is known to exist, but for which it may be necessary to follow a very long path step by step to find it. The name comes from this structure: *Polynomial Parity Arguments on Directed graphs*.

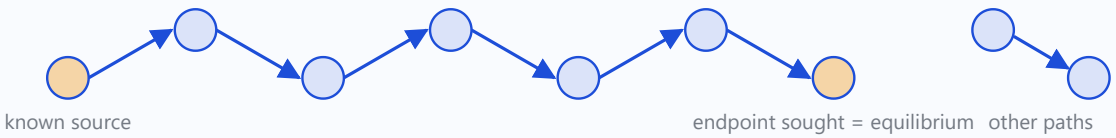


Figure 3.26: The path analogy for PPAD. A directed path that starts at a known source must end at some endpoint; the endpoint being sought is the equilibrium. The path can be exponentially long. There are other paths on the right as well.

We can read the analogy in Figure 3.26 as follows. The start of a path is known. How to get from each vertex to the next is easy to compute. The path cannot be infinite, because the number of vertices is finite; so the path certainly has an end. But the path may be exponentially long, and walking to its end may take astronomical time. The Lemke–Howson algorithm follows exactly such a path. Finding a Nash equilibrium is complete for this class, that is, it is among the hardest problems in it. This is why it is believed that equilibria in general games may not be efficiently findable. The situation also raises a conceptual question: if even a computer cannot find an equilibrium quickly, do real players "find" it? The power of the equilibrium concept as a predictive tool is tested by this question.

3.6.8 Approximate equilibria and computational tools

Definition 3.10

Let the payoffs be scaled to the interval [0, 1]. A profile is an **ε-Nash equilibrium** if and only if no player can increase their payoff by more than ε through a unilateral deviation.

The scaling matters: if payoffs are multiplied by 1000, every deviation gain is multiplied by 1000 too, and ε becomes meaningless for the same profile. On the interval [0, 1], $\varepsilon = 0.01$ means "no player can gain more than 1% of the largest possible payoff difference by deviating". Real-time systems (ad auctions, markets) often settle for this much.

Finding an exact equilibrium may be hard, but a profile in which "almost nobody wants to deviate" is often found quickly. Even simple profiles that mix a few strategies with equal probabilities give good approximations (Lipton, Markakis and Mehta, 2003). In two-player games a very simple profile is a ½-approximate equilibrium (Daskalakis, Mehta and Papadimitriou, 2006): pick a random row i of the row player; the column player plays the column j that is a best response to i; k is the row that is a best response to j; the row player mixes i and k half and half.

In practice, ready-made tools are used to compute equilibria. The Python library **Nashpy** offers support enumeration and the Lemke–Howson algorithm for two-player games. The inspection game is solved with Nashpy as follows:

```
import nashpy as nash
A = [[2, 2], [0, 4]]    # worker: Work / Shirk
B = [[1, 2], [-1, -4]]  # manager: Inspect / Don't
game = nash.Game(A, B)
for x, y in game.support_enumeration():
    print(x, y)  # [0.75 0.25] [0.5 0.5]
```

`A` is the worker's and `B` the manager's payoff matrix; the rows are Work/Shirk and the columns Inspect/Don't inspect. `nash.Game(A, B)` builds the game. `support_enumeration()` tries all pairs of supports and returns each equilibrium it finds as an (`x` , `y`) pair. In the output, `[0.75 0.25]` is the worker's probabilities (Work, Shirk), that is, $p^* = 3/4$; `[0.5 0.5]` is the manager's probabilities (Inspect, Don't inspect), that is, $q^* = 1/2$. Since the game has no pure equilibrium, there is no other output. The result is the same as the hand computation. A tool called **Gambit** does the same job (support enumeration, Lemke–Howson, linear programming) through an interface, and for games with many players.

There are also generalizations and refinements of the Nash equilibrium. The correlated equilibrium (Section 3.5.7) is a generalization. In games where moves are made in sequence, a narrower concept, the **subgame perfect equilibrium**, is used: it is found by backward induction in the game tree and rules out non-credible threats. Every subgame perfect equilibrium is a Nash equilibrium; the converse is false.

3.7 Discussion and checklist

The limits of the concept and the steps to follow in solving a game are gathered below.

Is an equilibrium fair? It need not be. An equilibrium is only a mutual best response. In the Battle of the Sexes the pure equilibria give unequal payoffs; in the Prisoner's Dilemma there is an outcome better for everyone. Fairness and equilibrium are different concepts. That an outcome is an equilibrium does not show that it is good or desirable; it shows only that it is self-sustaining.

Are people rational? Real decisions are affected by lack of information, time pressure, emotions, and social norms. Herbert Simon called this **bounded rationality**. That about half of the participants in one-shot Prisoner's Dilemma experiments cooperate is an indication of this. The model is still valuable, however: it offers a clear, precise point of comparison for studying strategic dependence. When real behavior departs from the model, the reason for the departure (trust, reputation, a sense of fairness) can be studied separately.

Ethical limits. Deception, manipulation, unfair advantage... Game theory does not answer the question "is every way of winning allowed?" The limits of strategic behavior are drawn by rules and institutions. Designing the rules so that the desired outcome is an equilibrium is the subject of **mechanism design**. The examples in this chapter carry the first traces of this idea: in the inspection game, raising the penalty changed the probability of inspection, and the traffic light created a correlated equilibrium without collisions.

Checklist. To solve a game step by step, follow the order below (Figure 3.27).

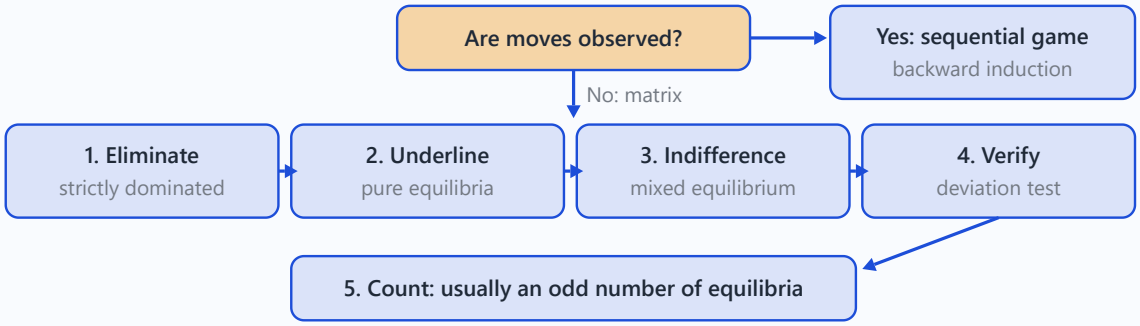


Figure 3.27: The steps of solving a game. If the moves are observed, the game is sequential and backward induction is used; if not, the steps on the matrix are elimination, underlining, indifference, verification, and counting.

1. **Read first.** Who is the row player and who is the column player? Whose number comes first in a cell? Are the numbers payoffs or costs (years, minutes)? If costs, convert to payoffs with a minus sign.
2. **Eliminate.** Iteratively delete strictly dominated strategies. Try mixtures as well.
3. **Underline.** In the remaining game, mark the best responses; cells with two marks are pure equilibria.
4. **Indifference.** Look for a mixed equilibrium with the indifference equations; check that the probabilities lie between 0 and 1.
5. **Verify.** Run the deviation test on every candidate.
6. **Count.** In games without ties the number of equilibria is usually odd; if an even number was found, an equilibrium may have been missed.
7. **Then write.** Write each equilibrium as a profile: for example (Middle, Left) = (3, 5). For a mixed equilibrium, give both players' probabilities and expected payoffs.

3.8 Summary

- A **Nash equilibrium** is a strategy profile in which no player, with the others held fixed, can raise their payoff by deviating alone: $u_i(s_i^*, s_{-i}^*) \geq u_i(s_i, s_{-i}^*)$. Equilibrium speaks of stability, not of efficiency or fairness (the Prisoner's Dilemma).

- A **best response** is the set of strategies that maximize the payoff against the others' choice. An equilibrium is a mutual best response. In matrix games, pure equilibria are found in $O(mn)$ time by marking the row player's largest payoff in each column and the column player's largest payoff in each row; in a tie, all the largest are marked.
- With continuous strategies, the best response is found with the derivative. The **Cournot** ($P = 12 - Y$) equilibrium is 4 units and a profit of 16 each; the cartel (3 each, 18 each) is not an equilibrium. In **Bertrand**, the price falls to cost and profit is zero.
- A **strictly dominated** strategy is a best response to no belief. Iterated strict elimination deletes no equilibrium and is independent of order; deleting weakly dominated strategies can lose equilibria. Mixtures can dominate too.
- In a **mixed equilibrium**, every pure strategy in the support gives the same expected payoff (the indifference principle). Hence a player's equilibrium probability is computed from the opponent's payoffs. In the inspection game $(p^*, q^*) = (3/4, 1/2)$, and when the penalty changes only q^* changes.
- **Nash's existence theorem**: every finite game has at least one mixed equilibrium (Kakutani / Brouwer fixed point). In nondegenerate games the number of equilibria is odd (Wilson).
- With **multiple equilibria**, the theory may not select a single outcome. Pareto dominance, risk dominance, focal points, and learning may suggest different choices (in Stag Hunt, (S, S) is Pareto dominant and (H, H) is risk dominant). Dominant-strategy equilibria are robust to mistakes; coordination equilibria are not.
- In a **correlated equilibrium**, obeying a mediator's recommendation is in everyone's interest. Because the obedience constraints are linear, it is found by linear programming; in the traffic-light example the payoff is 0, versus −1/10 at the mixed Nash equilibrium.
- In **zero-sum** games, max-min = min-max = v (von Neumann). If there is no saddle point, mixed strategies are needed (in the penalty kick $p^* = 4/11$, $v \approx 79.1\%$); the equilibrium is found by linear programming, and equilibria are interchangeable.
- **Complexity**: finding a mixed equilibrium in a general two-player game is PPAD-complete; in practice, support enumeration, Lemke–Howson, and ε-approximate equilibria are used.

Concept	What it says	How to find it
Nash equilibrium	Nobody wants to deviate alone	Mutual best response
Elimination of dominated strategies	A rational player does not play a dominated strategy	Iteratively delete strictly dominated strategies
Mixed equilibrium	Strategies in the support pay equally	Indifference equations
Existence (Nash 1950)	A finite game always has a mixed equilibrium	Kakutani / Brouwer fixed point
Multiple equilibria	The theory may not select a single outcome	Pareto / risk dominance, focal point
Correlated equilibrium	Obeying the mediator's recommendation is in everyone's interest	Obedience constraints + linear programming
Minimax	In zero-sum games max-min = min-max = v	Saddle point or LP

Concept	What it says	How to find it
Complexity	Exact equilibrium for two players, approximate equilibrium for many players, is PPAD-complete	Lemke–Howson, support enumeration

Table 3.21: The concepts of the chapter at a glance.

3.9 Key terms

Nash equilibrium pure Nash equilibrium strategy profile strategy set payoff function normal form unilateral deviation Pareto optimal best response best-response correspondence Cournot duopoly cartel Bertrand paradox strict dominance weak dominance dominant strategy dominant-strategy equilibrium IESDS common knowledge mixed strategy support expected payoff indifference principle inspection game Nash's existence theorem fixed-point theorem oddness theorem degenerate game multiple equilibria equilibrium selection Pareto dominance risk dominance focal point correlated equilibrium zero-sum game maximin minimax saddle point value of the game minimax theorem interchangeability security game linear programming support enumeration Lemke–Howson algorithm PPAD-complete ε-approximate equilibrium

3.10 Test Yourself

1. In the cells, (u_1, u_2): (Top, Left) = (3, 2), (Top, Right) = (1, 1), (Bottom, Left) = (0, 0), (Bottom, Right) = (2, 3). What are all the Nash equilibria of the game? What is the two players' expected payoff at the mixed equilibrium?

   Answer

   Row best responses: Left → Top (3 > 0), Right → Bottom (2 > 1). Column best responses: Top → Left (2 > 1), Bottom → Right (3 > 0). The pure equilibria are (Top, Left) and (Bottom, Right). Mixed equilibrium: the row player's indifference is $3q + 1(1 - q) = 0 \cdot q + 2(1 - q)$, that is, $1 + 2q = 2 - 2q$ and $q = P(\text{Left}) = 1/4$. The column player's indifference is $2p + 0 = 1 \cdot p + 3(1 - p)$, that is, $p = P(\text{Top}) = 3/4$. Expected payoffs: row $3 \cdot ¼ + 1 \cdot ¾ = 3/2$, column $2 \cdot ¾ = 3/2$. There are three equilibria in all (consistent with the odd-number rule).

2. The rows are Top, Middle, Bottom; the columns are Left, Right. (Top, Left) = (2, 1), (Top, Right) = (1, 3), (Middle, Left) = (3, 0), (Middle, Right) = (2, 1), (Bottom, Left) = (1, 2), (Bottom, Right) = (0,

0). What remains when strictly dominated strategies are deleted iteratively?

Cevap

Middle beats Top in every column (3 > 2, 2 > 1): Top is deleted. Middle also beats Bottom (3 > 1, 2 > 0): Bottom is deleted. With only the Middle row left, the column player gets 0 in Left and 1 in Right: Left is deleted. The remaining (Middle, Right) = (2, 1) is the game's only Nash equilibrium. Note: at the start there is no domination among the columns (in the Top row Right is 3 > 1, but in the Bottom row Left is 2 > 0).

3. The row player's payoffs: Top (Left 5, Right 0), Middle (2, 2), Bottom (0, 5). Does a pure strategy dominate Middle? For which values of p does the strategy that mixes Top with probability p and Bottom with probability $1 - p$ strictly dominate Middle?

Cevap

No pure strategy dominates it: Middle is better than Bottom in Left (2 > 0) and better than Top in Right (2 > 0). The mixture gives $5p$ against Left and $5(1 - p)$ against Right. Strict domination requires $5p > 2$ and $5(1 - p) > 2$, that is, $2/5 < p < 3/5$. For example, the ½–½ mixture gives 2.5 in every column.

4. In a Cournot duopoly the price is $P = 18 - (y_1 + y_2)$ and the cost is 0. What are the quantities, price, and profits at the Nash equilibrium? If the firms form a cartel that splits production equally, what does each earn, and what does a firm that deviates from the cartel earn?

Cevap

Profit $y_1(18 - y_1 - y_2)$; derivative $18 - 2y_1 - y_2 = 0 \Rightarrow BR_1(y_2) = (18 - y_2)/2$. Symmetric equilibrium: $2y = 18 - y \Rightarrow y^* = 6$, price 6, profit 36 each. The cartel maximizes total $Y(18 - Y)$ at $Y = 9$: 4.5 each, price 9, profit 40.5 each. While the rival produces 4.5, the best response is $(18 - 4.5)/2 = 6.75$, and the deviating firm earns $6.75 \cdot 6.75 = 45.5625$. The cartel is not an equilibrium.

5. In a zero-sum game the row player's payoff matrix is $A = [[5, 1], [2, 4]]$. What are the pure maximin and minimax? If there is no saddle point, what are the two players' minimax strategies and the value of the game?

Cevap

Row minima 1, 2 ⇒ maximin 2; column maxima 5, 4 ⇒ minimax 4. $2 < 4$, so there is no saddle point. If the row player plays the first row with probability p, they get $5p + 2(1 - p) = 2 + 3p$ and $p + 4(1 - p) = 4 - 3p$ against the columns; equality gives $p = 1/3$, $v = 3$. If the column player plays the first column with probability q, $5q + 1(1 - q) = 2q + 4(1 - q) \Rightarrow q = 1/2$, which again gives 3.

6. A single guard protects one of two targets. Target H has value 5 and target L has value 3; an attacker who hits the unprotected target gains its value, and one who hits the protected target gets 0. With what probability should the guard protect H, and what does the attacker gain? What would happen if the guard always protected H?

   Cevap

   To leave the attacker indifferent: $5(1 - p) = 3p \Rightarrow p = 5/8$. The attacker's payoff is $3 \cdot 5/8 = 15/8 \approx 1.9$. If the guard always protected H, the attacker would hit L and get 3.

7. In a version of Stag Hunt, hunting stag together pays 5 each, hare pays 2 in every case, and going for the stag alone pays 0. What is the stag probability at the mixed equilibrium? Which equilibrium is Pareto dominant and which is risk dominant?

   Cevap

   If the partner chooses Stag with probability q, Stag gives $5q$ and Hare gives 2: $q = 2/5$. (S, S) is Pareto dominant (5 > 2). The product of deviation losses is $(5 - 2)^2 = 9$ at (S, S) and $(2 - 0)^2 = 4$ at (H, H); (S, S) is risk dominant too. In this version the two criteria select the same equilibrium; a partner who shows up with only 40% probability is enough to make stag worth choosing.

8. In the Chicken game at the intersection ((Swerve, Swerve) = (0, 0), (Swerve, Straight) = (−1, 1), (Straight, Swerve) = (1, −1), (Straight, Straight) = (−10, −10)), a light recommends (Swerve, Straight) or (Straight, Swerve) with probability one half each. Is this a correlated equilibrium? What would happen if the light recommended (Swerve, Swerve) or (Straight, Straight) with probability one half each?

   Cevap

   The first light is a correlated equilibrium. A driver who is recommended "Swerve" knows the opponent has certainly been told "Straight": obeying gives −1, deviating gives −10. A driver who is recommended "Straight" knows the opponent has certainly been told "Swerve": obeying gives 1, deviating gives 0. The expected payoff is 0. The second light is not a correlated equilibrium: a driver who is recommended "Straight" knows the opponent will also go straight; obeying gives −10, deviating to Swerve gives −1, and the driver deviates.

3.11 Further reading

- M. J. Osborne, *An Introduction to Game Theory* (2004), Chapters 2–4: covers the Nash equilibrium, mixed strategies, and dominance with many worked examples and little mathematics.

- T. Roughgarden, *Twenty Lectures on Algorithmic Game Theory* (2016), Lectures 13, 18, and 20: presents the hierarchy of equilibrium concepts, the proof of the minimax theorem via learning, and PPAD from a computer-science viewpoint.
- A. R. Karlin and Y. Peres, *Game Theory, Alive* (2017): gives zero-sum games and the fixed-point proof of Nash's theorem in detail at the student level.
- C. Daskalakis, P. Goldberg, and C. Papadimitriou, "The complexity of computing a Nash equilibrium" (2009): the journal version of the paper that shows the PPAD-completeness of finding a Nash equilibrium; its introduction can be read without technical detail.

3.12 References

1. R. J. Aumann, “Subjectivity and correlation in randomized strategies”, *Journal of Mathematical Economics* 1(1), 67–96, 1974.
2. X. Chen, X. Deng, “Settling the complexity of two-player Nash equilibrium”, *IEEE FOCS*, 2006.
3. V. Conitzer, T. Sandholm, “New complexity results about Nash equilibria”, *Games and Economic Behavior* 63(2), 621–641, 2008.
4. C. Daskalakis, P. Goldberg, C. Papadimitriou, “The complexity of computing a Nash equilibrium”, *SIAM J. Computing* 39(1), 2009 (conference version *ACM STOC*, 2006).
5. C. Daskalakis, A. Mehta, C. Papadimitriou, “A note on approximate Nash equilibria”, *WINE*, LNCS 4286, 297–306, 2006.
6. D. Easley, J. Kleinberg, *Networks, Crowds, and Markets*, Cambridge, 2010 (Chapter 6: Games).
7. K. Etessami, M. Yannakakis, “On the complexity of Nash equilibria and other fixed points”, *SIAM J. Computing* 39(6), 2010 (conference version *IEEE FOCS*, 2007).
8. I. Gilboa, E. Zemel, “Nash and correlated equilibria: Some complexity considerations”, *Games and Economic Behavior* 1(1), 80–93, 1989.
9. J. C. Harsanyi, R. Selten, *A General Theory of Equilibrium Selection in Games*, MIT Press, 1988.
10. A. R. Karlin, Y. Peres, *Game Theory, Alive*, AMS, 2017.
11. R. J. Lipton, E. Markakis, A. Mehta, “Playing large games using simple strategies”, *ACM EC*, 2003.
12. J. F. Nash, “Equilibrium points in n-person games”, *PNAS* 36(1), 1950.
13. J. F. Nash, “Non-cooperative games”, *Annals of Mathematics* 54(2), 1951.
14. N. Nisan, T. Roughgarden, É. Tardos, V. Vazirani (ed.), *Algorithmic Game Theory*, Cambridge, 2007 (Chapters 1–3).
15. M. J. Osborne, *An Introduction to Game Theory*, Oxford University Press, 2004 (Chapters 2–4).
16. I. Palacios-Huerta, “Professionals play minimax”, *Review of Economic Studies* 70(2), 395–415, 2003.
17. C. H. Papadimitriou, “On the complexity of the parity argument and other inefficient proofs of existence”, *Journal of Computer and System Sciences* 48(3), 498–532, 1994.
18. T. Roughgarden, *Twenty Lectures on Algorithmic Game Theory*, Cambridge, 2016 (Lectures 13, 18, and 20).
19. A. Rubinstein, “Inapproximability of Nash equilibrium”, *ACM STOC*, 2015.

20. D. Sally, “Conversation and cooperation in social dilemmas: a meta-analysis of experiments from 1958 to 1992”, *Rationality and Society* 7(1), 1995.

21. T. C. Schelling, *The Strategy of Conflict*, Harvard University Press, 1960.

22. M. Tambe, *Security and Game Theory*, Cambridge University Press, 2011.

23. J. von Neumann, “Zur Theorie der Gesellschaftsspiele”, *Mathematische Annalen* 100, 1928.

24. R. Wilson, “Computing equilibria of n-person games”, *SIAM J. Applied Mathematics* 21, 1971.

PART II

Mechanisms

Chapters 4–7

Mechanism Design

Game theory usually takes the rules of a game as given and predicts what the players will do in equilibrium. Mechanism design turns the question around: the desired outcome is fixed, and the task is to find rules that produce it. The difficulty is that players can hide private information such as values, preferences, or costs from the designer. These notes start with direct mechanisms and the revelation principle, then move on to incentive compatibility, individual rationality, and Myerson's lemma. Next comes VCG, the most important general mechanism that uses money, followed by the Gale–Shapley, median, Boston, and TTC mechanisms for settings where money is not available. The last topic is Bayesian games with incomplete information.

LEARNING OUTCOMES

Concepts
Explain social choice functions, direct mechanisms, the revelation principle, and Bayes–Nash equilibrium.

Incentive analysis
Test whether a mechanism is incentive compatible (IC) and individually rational (IR).

VCG computation
Compute the efficient outcome and each player's VCG payment from the welfare that others would obtain "without that player".

Design without money
Find stable matchings with Gale–Shapley and trades with TTC; show why the median mechanism is strategyproof.

4.1 Basic concepts

The question of this chapter is the following. If everyone in a group pursues their own interest, and each person knows something the others do not, what rules let us reach a good decision for the group? A small bargaining game shows how rules shape behavior.

4.1.1 A first game: the Ultimatum Game

Two players must split 10 tokens. The first player is the **proposer** and the second is the **responder**. The proposer offers the responder a share, for example "3 tokens for you, 7 for me". The responder either accepts or rejects. If the responder accepts, the split is carried out. If the responder rejects, both get 0. There is no haggling: the offer is made once. The game takes its name from this feature, since the offer is an "ultimatum", take it or leave it.

Figure 4.1 shows the structure of the game. The proposer (P) first chooses a number x, the number of tokens offered to the responder. The responder (R) then sees this number and decides. If the offer is accepted, the payoffs are $(10 - x, x)$, where the first number is the proposer's payoff and the second is the responder's. If it is rejected, the payoffs are $(0, 0)$.

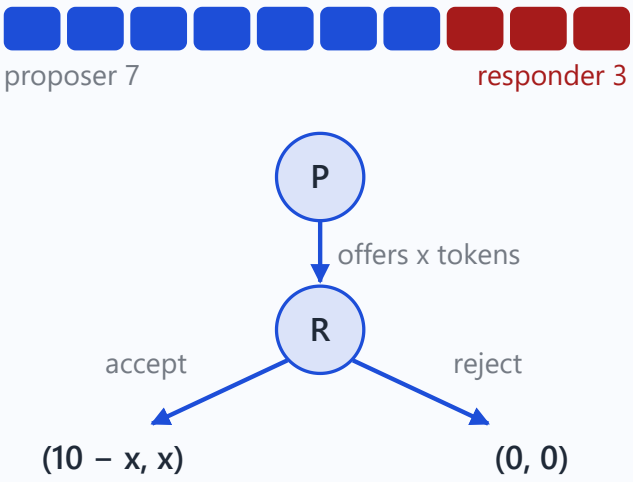


Figure 4.1: The Ultimatum Game. The proposer (P) offers the responder x tokens; if the responder (R) accepts, the outcome is $(10 - x, x)$, and if R rejects, it is $(0, 0)$. The bar at the top shows a 7–3 split.

Two questions arise. Should the responder accept an offer of 1 token? How many tokens should the proposer offer? Both answers depend on what the player cares about. For a responder who cares only about the number of tokens, 1 token is better than 0, so rejecting looks irrational. Yet many people treat an offer of 1 token as an insult and reject it. This tension brings out the central question of the chapter: how do rules affect players' behavior, and how should a rule maker, taking that effect into account, choose the rules?

The Ultimatum Game is a small **mechanism**: the proposer presents the responder with the rule "this split or nothing". The game-theoretic solution is given in 4.2.5.

4.1.2 What is mechanism design?

In the usual approach of game theory, the game is given: the players, their strategies, and their payoffs are known. The question is what will happen in this game. **Mechanism design** reverses this approach. The game is not given; the designer builds it. The designer has a desired outcome and looks

for rules that produce it. The rules determine two things: who gets what (the allocation) and who pays what (the payments).

This task is harder than it first appears, because the designer does not know the players' private information. A seller who wants to give an item to the person who values it most does not know who values it how much. A municipality does not know how much residents value a park. This information is held only by the players. The designer has to ask the players for it, and the players may lie if their interests call for it.

The goal of mechanism design is therefore this: to reach the desired outcome even when players are selfish and strategic, that is, even when they pursue their own interests. The designer cannot order the players to "tell the truth"; the designer can only choose the rules. The rules must be chosen so that telling the truth, or behaving as desired, is in the player's own interest.

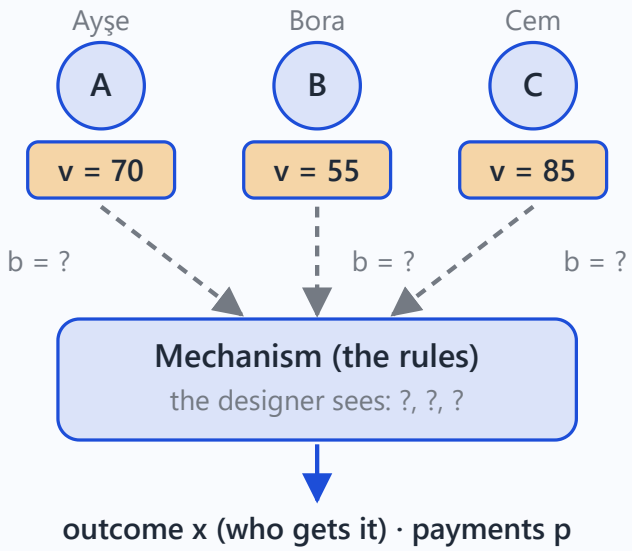


Figure 4.2: The basic situation in mechanism design. The values of Ayşe, Bora, and Cem (70, 55, 85) are known only to themselves; the mechanism sees only the reports ($b = ?$) and computes the outcome x and the payments p from them.

Figure 4.2 illustrates this situation. Three people value an item at 70, 55, and 85, respectively. These numbers are in the players' heads; the designer cannot see them ("the designer sees: ?, ?, ?"). The mechanism computes the outcome and the payments from the numbers b that the players report. The reports need not equal the true values. In a good mechanism, reporting the true value is in the player's own interest.

The importance of the field has also been recognized with scientific prizes. The 2007 Nobel Prize in Economics went to Leonid Hurwicz, Eric Maskin, and Roger Myerson "for having laid the foundations of mechanism design theory". In 2012 Alvin Roth and Lloyd Shapley were honored for their work on stable matching and market design.

4.1.3 Analysis and synthesis

Game theory and mechanism design are the same road traveled in opposite directions. Game theory is a tool of **analysis**: the rules are given, the players' behavior (their equilibrium strategies) is found, and the outcome that follows from that behavior is predicted. Mechanism design is a tool of **synthesis**: the desired outcome (the goal) is given, and we look for rules under which the players will behave in a way that leads to that goal.

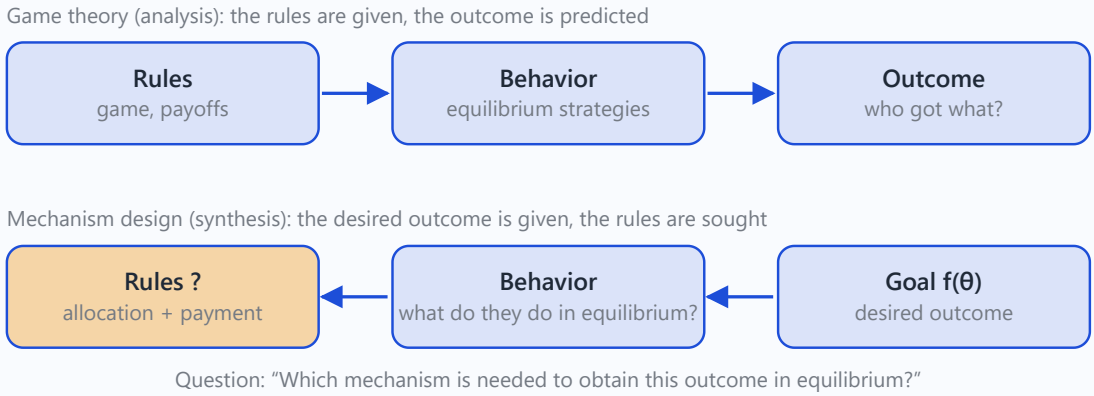


Figure 4.3: Analysis and synthesis. In the top row, game theory goes from rules to behavior and from behavior to outcome. In the bottom row, mechanism design starts from the goal $f(\theta)$, asks what will be done in equilibrium, and finds rules made up of an allocation and a payment.

Figure 4.3 puts the two directions side by side. The top row asks: "What will the players do under these rules, and who will win?" The bottom row asks: "Which mechanism is needed to obtain this outcome in equilibrium?" In short, in game theory the flow runs from rules to outcome; in mechanism design it runs from outcome to rules.

Working in the direction of synthesis has an important consequence. The designer cannot force the players; the players themselves decide which strategy to play. The designer's only tool is the rules. Every proposal in mechanism design is therefore tested with an equilibrium question: under the proposed rules, will the players actually behave as intended? If they do not, the goal is not reached, however well meant the rules may be.

4.1.4 Incentives

Mechanism design rests on a simple observation: people respond to **incentives**. If the rules make a certain behavior profitable, players move toward that behavior. If a parking fee goes up, people drive less. If wrong answers lose points on an exam, students leave the questions they are unsure about blank. In every case the rule maker should ask: "Which behavior does this rule reward?"

A good mechanism rewards honest behavior and deters strategic lying. A mechanism in which telling the truth is in the player's own interest is called **incentive compatible**. The formal definition of the concept is in 4.2.4. Two familiar examples show what incentive incompatibility leads to.

The free-rider problem. Suppose a park is to be built in a neighborhood and its cost is to be shared according to how much each resident values the park. Each resident states for themselves how much they value it. A resident who understates the value pays less, and if the park is built anyway, enjoys it just as much. Everyone is then inclined to understate. A person who wants to benefit from what others pay is called a **free rider**. If everyone acts as a free rider, the reports in total may not cover the cost, and the park that everyone wants is never built.

Strategic voting. In an election, a voter whose favorite candidate has no chance of winning may vote for a second choice who does. This is the voter hiding their true preference. The rule (the candidate with the most votes wins) rewards this behavior. As a result, the election outcome may fail to reflect the voters' true preferences.

The same chain operates in both examples: incentive incompatibility pushes players to give false information, and false information leads to inefficient or unfair outcomes. Mechanism design tries to break this chain at the start: giving true information must be in the player's own interest.

> **Common mistake**
>
> "If players are honest, any rule works well, so let us simply assume honesty." This is exactly the assumption that mechanism design does not make. It accepts that players will pursue their own interests and chooses the rules so that they are robust to this behavior. Honest behavior is not an assumption; it is the result of a good mechanism.

4.1.5 Real-world applications

The problems of mechanism design appear in many areas of everyday life. In all the examples below, an institution makes decisions that depend on the private information of individuals.

- **Tax systems.** A person's income is private information. The government cannot see income directly; it relies on declarations and audits. Tax rates and audit rules should encourage honest declarations.
- **Public goods financing.** For goods that everyone benefits from, such as parks, roads, or libraries, there are two questions: should the project be built, and who should pay how much? Both questions depend on residents' hidden values. The mechanism-based solution is VCG in 4.3.
- **Procurement and auctions.** Public purchases are made through tenders; spectrum licenses and works of art are sold at auction. The seller does not know the highest price the buyers are willing to pay; the rules try to bring this information out.
- **Online advertising.** A search engine runs an auction for ad slots on every query, within milliseconds. The value an advertiser places on a click is private.
- **Matching markets.** In school placement, medical residency matching, and dorm room exchange, money is not used, but the rules still matter. Students' preference lists are private information, and the rules should reward writing honest lists (4.4).

- **Grading group projects.** If students report their own contributions to a group project, the grading rule should discourage exaggeration.

The common structure of these examples is the basis of the formal model: the players, the players' private information (types), the possible outcomes, and the designer's goal. These four elements are defined in 4.2.1.

4.2 Direct mechanisms

The formal language of mechanism design rests on four concepts: a player's private information (the type), the designer's goal (the social choice function), the direct mechanism in which players report only their types, and the revelation principle, which justifies focusing on such mechanisms. Whether a mechanism is good is tested against the criteria of incentive compatibility, individual rationality, efficiency, and budget balance. In single-parameter settings, Myerson's lemma characterizes incentive-compatible mechanisms exactly.

4.2.1 Types and the social choice function

A mechanism has n players, numbered $1, 2, \ldots, n$. Each player has private information known only to that player: the value placed on an item, a ranking of the alternatives, or the cost of doing a job. This private information is called the player's **type**.

Definition 4.1

The **type** θ_i of player i is an element of the set of possible types Θ_i: $\theta_i \in \Theta_i$. The type is private information. The list of all players' types, $\theta = (\theta_1, \ldots, \theta_n)$, is called the **type profile**. The set of possible outcomes is denoted O. A **social choice function**

$$f : \Theta_1 \times \ldots \times \Theta_n \to O$$

is a rule that assigns a "desired" outcome to each type profile.

The symbols in the definition may look unfamiliar at first. Θ is the capital form of the Greek letter "theta" and denotes a set; the lowercase θ is an element chosen from that set. The sign $\times$ in $\Theta_1 \times \ldots \times \Theta_n$ is the Cartesian product: the set of all lists that can be formed by choosing one type from each player. For example, if each of two players can be of type "low" or "high", there are four type profiles: (low, low), (low, high), (high, low), (high, high). The social choice function assigns an outcome to each of these four profiles.

The outcome set O depends on the problem. If a single item is being sold, the outcome answers the question "who gets the item?": $O = \{\text{Ayşe, Bora, Cem, nobody}\}$. If a municipality is choosing one of several projects, the outcome is the chosen project: $O = \{\text{Park, Library, Gym}\}$. In a school placement,

the outcome is an assignment that says which school each student attends.

The social choice function expresses the designer's goal. The designer answers the question "if I knew the types, what would I want to do?" The most common goal is **efficiency**: the outcome should maximize the sum of the players' values. In selling a single item, this means giving it to the player who values it most.

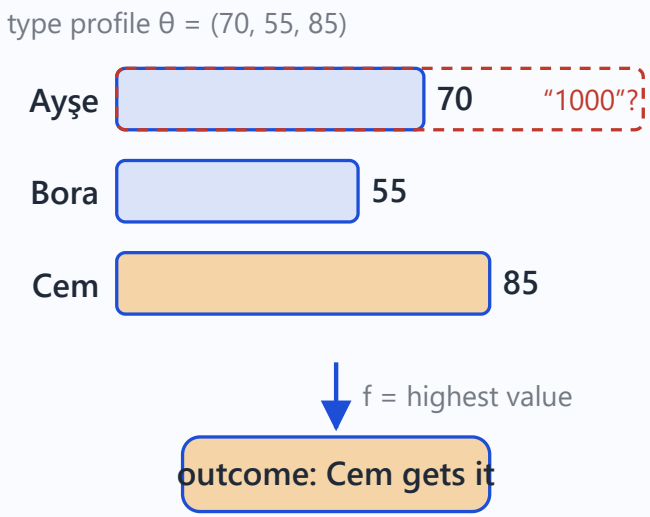


Figure 4.4: The efficient social choice function. At the type profile $\theta = (70, 55, 85)$ the highest value is Cem's, so $f(\theta)$ = "Cem gets it". The dashed box recalls the case in which Ayşe could report her value as "1000".

Example 4.1

Ayşe, Bora, and Cem value an item at 70, 55, and 85, respectively. The type profile is $\theta = (70, 55, 85)$. The efficient social choice function is the rule "the item goes to the player with the highest value".

1. Compare the values: $85 > 70 > 55$.
2. The largest value is Cem's; $f(\theta)$ = "Cem gets it".
3. The social welfare of this outcome, that is, the value of the person who gets the item, is 85. Had the item gone to Ayşe the welfare would have been 70, and had it gone to Bora, 55. The efficient outcome picks the largest of these three.

The problem is that the designer cannot see these three numbers and has to ask the players. If the rule is only "whoever reports the highest value gets it" and there is no payment, every player says "my value is 1000". The reports then carry no information, and f cannot be applied.

The example brings out the central question of mechanism design: are there rules that implement f even when the players pursue their own interests? Sometimes the answer is yes and sometimes no. For selling a single item the answer is yes: it is enough to choose the price the winner pays correctly

(4.2.4). In some settings, however, no rule can achieve everything we want at once (4.2.7).

4.2.2 Direct mechanisms and quasi-linear utility

The designer has to get the information from the players. The most direct way is to ask each player for their type; mechanisms of this kind are called direct mechanisms.

Definition 4.2

In a **direct mechanism**, each player i submits a **report** $b_i \in \Theta_i$. The report may or may not equal the true type. From the report profile $b = (b_1, \ldots, b_n)$ the mechanism computes an outcome $x(b)$ and a payment $p_i(b)$ for each player. The player's **quasi-linear utility** is

$$u_i = v_i(x) - p_i$$

Here $v_i(x)$ is the value that a player of type θ_i places on outcome x. With a single item, $v_i = \theta_i$ if the player gets the item and 0 otherwise.

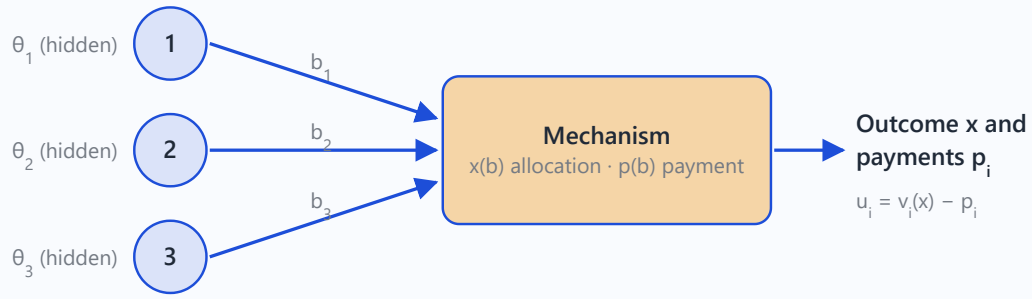


Figure 4.5: A direct mechanism. The types $\theta_1, \theta_2, \theta_3$ of three players are hidden; each player sends a report b_i. From the reports the mechanism computes the allocation $x(b)$ and the payment $p(b)$; the player's utility is $u_i = v_i(x) - p_i$.

Figure 4.5 shows a direct mechanism as a box. The input to the box is the reports and its output is the outcome and the payments. Everyone knows the rules inside the box, and the players decide what to report accordingly.

What does "quasi-linear" mean? The money part of the utility is linear: paying 1 TL more reduces utility by exactly 1 unit. The value part $v_i(x)$ can be any function. This assumption ensures that value and money are measured in the same unit. A player who gets an item worth 85,000 TL for 70,000 TL has utility 15,000 TL; a player who gets nothing and pays nothing has utility 0. Quasi-linear utility is the standard model for settings in which money can be transferred between people; all the mechanisms with payments in this chapter rest on this assumption.

Example 4.2

Let the values of Ayşe, Bora, and Cem be 70, 55, and 85. Suppose a direct mechanism uses this rule: the highest report wins, the winner pays their own report, and the losers pay nothing.

1. If everyone reports honestly, $b = (70, 55, 85)$. Cem wins and pays 85. Cem's utility is $85 - 85 = 0$; Ayşe's and Bora's utilities are 0.
2. If Cem reports 71, his is still the highest report. Cem pays 71 and his utility is $85 - 71 = 14$.
3. By lying, Cem gained 14 units. In this mechanism an honest report gains Cem nothing, while a lie pays off.

If the rule stays the same and only the payment is changed to "the winner pays the second-highest report", the situation is different. Whether Cem reports 85 or 71, he pays 70 and his utility is 15. Since the payment does not depend on Cem's own report, changing the report does not help him. The graph of this difference is in 4.2.4.

Indirect mechanisms. Not every mechanism asks players for their types. In an English auction the price rises slowly, and players show by raising a hand whether they are still in. In a negotiation the parties make offers in turn. Mechanisms of this kind are called **indirect mechanisms**: the player's strategy is not to state a type but to take some other action (raising a hand, making an offer, writing down a ranking). Indirect mechanisms are very common in practice, but in theory they can be reduced to direct mechanisms. This reduction is provided by the revelation principle.

4.2.3 The revelation principle

The mechanism designer faces infinitely many possible rules: auctions, sequential offers, complicated voting procedures, and so on. Examining all of them is impossible. The **revelation principle** narrows this search space considerably: it is enough to look at direct mechanisms in which honest reporting is an equilibrium.

Theorem 4.1 (Revelation principle)

If an outcome can be implemented as an equilibrium of an indirect mechanism M, then, under suitable conditions, a direct mechanism M^* in which honest reporting is an equilibrium can implement the same outcome.

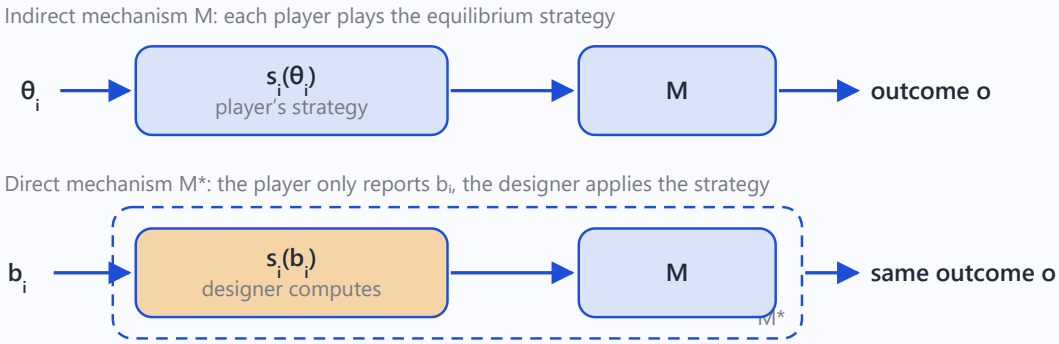


Figure 4.6: The idea of the revelation principle. At the top, in the indirect mechanism M, each player plays the equilibrium strategy $s_i(\theta_i)$ for their type. At the bottom, in the direct mechanism M^*, the player only reports b_i; the designer computes $s_i(b_i)$ on the player's behalf and feeds it to M. The outcome is the same.

The idea behind the principle is shown in Figure 4.6. Let M be an indirect mechanism and let the players' equilibrium strategies be $s_1, \ldots, s_n$. Each strategy is a rule: a player of type θ_i chooses the action $s_i(\theta_i)$. The designer builds the following new mechanism: "Tell me your type; I will play, on your behalf, the equilibrium strategy you would play at that type." This new mechanism M^* is direct, because it asks the player only for their type.

Proof. Here is why honest reporting is an equilibrium in M^*. Let the other players report honestly. Then the designer plays $s_j(\theta_j)$ on their behalf, which is their equilibrium behavior in M. If player i reports θ_i honestly, the designer plays $s_i(\theta_i)$ for that player. If player i reports some other b_i, the designer plays $s_i(b_i)$ for that player; this is the same as a player of type θ_i in M choosing a different action instead of the equilibrium strategy. Since s is an equilibrium, such a deviation is not profitable in M. So lying is not profitable in M^* either. Hence honest reporting is an equilibrium in M^*, and under honest reporting M^* produces the equilibrium outcome of M. □

The consequence of this simple observation is powerful: to answer the question "is there a mechanism that implements this goal?", it is enough to look at direct mechanisms in which honest reporting is an equilibrium. If there is no such direct mechanism, there is no indirect mechanism either.

Example 4.3

In an English auction the price starts at zero and rises slowly. Each bidder stays in until the price reaches their own value and then drops out. Let the values be 70, 55, and 85.

1. When the price reaches 55, Bora drops out, because winning at a higher price would give negative utility.
2. When the price reaches 70, Ayşe drops out. Only Cem remains.
3. Cem gets the item for about 70, that is, the second-highest value. His utility is $85 - 70 = 15$.

Now let the designer say to the players: "Write your value on a piece of paper; I will stay in on your behalf until the price reaches your value." This is a direct mechanism. When the papers are opened, the highest value wins and pays the second-highest value. This is the second-price (Vickrey) auction. By the revelation principle, writing the true value is best in this direct mechanism, because writing a wrong value is the same as dropping out at the wrong moment in the English auction.

The principle also has limits. The revelation principle does not say that "honest reporting is a dominant strategy in every mechanism". It says only that honest reporting is *an equilibrium* in the constructed mechanism M^*. What kind of equilibrium it is depends on the equilibrium concept used in M. If the strategies in M are dominant strategies, honesty is a dominant strategy in M^*. If the strategies in M form only a Bayes–Nash equilibrium (4.5.1), honesty in M^* is also only a Bayes–Nash equilibrium. These two cases are defined below with two separate incentive conditions.

Common mistake

"By the revelation principle direct mechanisms are better, so they should always be used in practice." The principle is an existence and reduction result; it does not say that a direct mechanism is better in practice. Indirect mechanisms may be easier to understand, may ask for less information, or may protect players' privacy. In an English auction the winner never reveals their own value; in a second-price auction they have to write it down.

4.2.4 Incentive compatibility and individual rationality

Two conditions are required for a direct mechanism to work well. First, telling the truth must be in the player's own interest. Second, the player must be willing to participate in the mechanism. The first condition is called incentive compatibility, the second individual rationality.

To make the notation easier, we write $u_i(\theta_i; b_i, b_{-i})$. This expression means: player i with true type θ_i reports b_i, the others report b_{-i}, and this is the player's true utility. b_{-i} means "the reports of everyone other than i". The θ_i before the semicolon shows which true value the utility is measured against, and what comes after it shows the reports.

Definition 4.3

A direct mechanism is **dominant-strategy incentive compatible** (DSIC) if, for every player i, every true type θ_i, every lie b_i, and every report profile b_{-i} of the others, the following inequality holds:

$$u_i(\theta_i;\, \theta_i,\, b_{-i}) \geq u_i(\theta_i;\, b_i,\, b_{-i})$$

That is, whatever the others report, telling the truth is at least as good as lying.

DSIC is a very strong condition. The player need not know or guess anything about the others. Whether the others are honest or lying, smart or mistaken, an honest report is best for the player; that is, it is a dominant strategy. DSIC mechanisms therefore give players simple advice: "Say what you think, and do not worry about the rest."

Definition 4.4

Suppose types are drawn from a probability distribution known to everyone. A direct mechanism is **Bayesian incentive compatible** (BIC) if, when the other players report honestly, telling the truth maximizes each player's expected utility. The expectation is taken over the others' unknown types.

The BIC condition is weaker than DSIC in two ways. First, it requires that telling the truth be best only when the others are *honest*; it says nothing if the others lie. Second, it requires utility to be maximal not in every case but *on average* (as an expected value). The expected value is the sum of the utility in each possible case multiplied by the probability of that case. For example, if the rival's value is 4 or 8 with equal probability, and with some report the player gets 6 in the first case and 0 in the second, the expected utility is $\frac{1}{2} \cdot 6 + \frac{1}{2} \cdot 0 = 3$.

Theorem 4.2 (DSIC ⇒ BIC)

Every DSIC mechanism is BIC. The converse is not true in general.

Proof. Let the mechanism be DSIC. Whatever the others report, an honest report gives player i at least as much utility as lying. When the others are honest, their reports are their types, and this is one of the "any b_{-i}" cases. For every type profile, the honest utility is known to be at least the lying utility. If these inequalities are multiplied by the same probabilities and summed, the inequality is preserved. Hence an honest report is also best in expected utility. □

The first-price auction shows why the converse fails (4.5.4).

Definition 4.5

A mechanism is **individually rational** (IR) if participation gives no player lower utility than staying out of the system. If the utility of staying out is taken to be 0, then under honest reporting every player must satisfy

$$u_i(\theta_i;\, \theta_i,\, b_{-i}) \geq 0$$

.

The IR condition is about participation. If the mechanism cannot force people to take part, participants must not lose out. If participating makes the player worse off, the player stays out, and the mechanism never gets their information. For example, if the winner of an auction is forced to pay more than their value, IR is violated.

Incentive compatibility is easiest to see with a numerical example. Suppose an item is sold between two bidders. Let the bidder's value be 10 and the rival's bid 6. Two payment rules are compared. Under the **first-price** rule the winner pays their own bid. Under the **second-price** rule the winner pays the rival's bid. In both rules the highest bid wins.

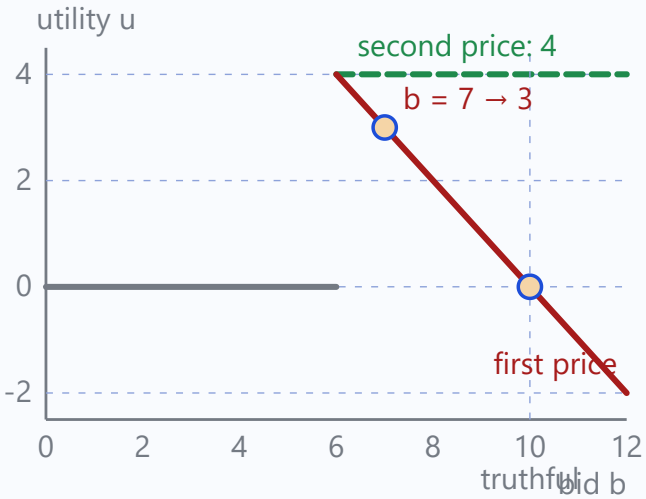


Figure 4.7: Value 10, rival bid 6. The horizontal axis is the bid b and the vertical axis is the utility u. Under first price (red), utility is $10 - b$, which falls to 0 at the truthful bid $b = 10$; a bid of 7 gives 3. Under second price (green, dashed), every bid above 6 gives 4.

Example 4.4

1. **A bid below 6.** Under both rules the bidder loses; the utility is 0.

2. **First price, bid b above 6.** The bidder wins and pays b. The utility is $10 - b$. The truthful bid $b = 10$ gives utility $10 - 10 = 0$. A bid of $b = 7$ gives $10 - 7 = 3$. Bidding below one's value pays off; the rule is not IC.
3. **Second price, bid above 6.** The bidder wins and pays the rival's bid, 6. The utility is $10 - 6 = 4$. Whether the bid is 7, 10, or 50, the utility is the same: 4.
4. **Second price, a lower bid.** If the bid drops below 6, the bidder loses and gets 0 instead of 4. The truthful bid (10) is never worse.

The red line in the graph shows that under the first-price rule the bidder wants to move away from truthfulness: the closer the bid gets to 6, the more the bidder gains. The green line shows that under second price the bid determines whether the bidder wins, but does not change the price paid.

The rival bid of 6 in the example is an arbitrary choice. Whatever the rival bid r, the same logic works under the second-price rule. If $r < 10$, the truthful bid wins and gives utility $10 - r > 0$; a lower bid either gives the same utility or loses. If $r > 10$, the truthful bid loses and gives 0; a bid above r wins but gives utility $10 - r < 0$. In every case the truthful bid is best; the second-price rule is DSIC. Moreover, the winner never pays more than their value and the loser pays nothing; the rule is IR.

Common mistake

Confusing IC with IR. IC is about *what the player reports*: is telling the truth better than lying? IR is about *whether the player participates*: is taking part better than staying out? A mechanism can satisfy one without the other. For example, add to the second-price auction a rule that everyone first pays a 3 TL entry fee. Since the fee does not depend on the bid, the truthful bid is still best (IC is preserved). But a winner with value 5 and a rival bid of 4 has utility $5 - 4 - 3 = -2$, and the losers' utility is −3: IR is violated.

4.2.5 Back to the Ultimatum Game

With incentive compatibility and individual rationality defined, the Ultimatum Game can be revisited. In the game, the proposer presents the responder with a "take it or leave it" mechanism. The responder's IR condition is that what they get from accepting is no less than what they get from rejecting.

The theoretical answer. If the responder cares only about the number of tokens, accepting an offer of x tokens gives x and rejecting gives 0. As long as $x \geq 0$, accepting is the best response; this is the responder's IR condition. At $x = 0$, accepting and rejecting both give 0, so the responder is indifferent; both options are best responses.

The proposer anticipates this behavior and chooses the offer accordingly. Because the game is sequential, the solution is found by backward induction: first determine what the responder would say to each offer, then let the proposer choose, knowing that. The equilibria found this way are called **subgame perfect equilibria**. If offers are whole numbers, there are two subgame perfect equilibria in pure strategies.

Example 4.5

1. **The responder also accepts 0.** The responder then accepts every offer. To maximize their own share $10 - x$, the proposer offers $x = 0$. The outcome is (10, 0).
2. **The responder rejects 0.** Since the responder is indifferent, rejecting 0 is also a best response. Offers $x \geq 1$ are then accepted and 0 is rejected. If the proposer offers 0 they get 0; if they offer 1 they get 9. The best is $x = 1$ and the outcome is (9, 1).
3. There is no other equilibrium in pure strategies: since the responder always accepts 1 and above, the proposer does not want to offer more than 1. If mixed strategies are counted too, the responder may accept 0 with probability p; the proposer earns $10p$ from offering 0 and 9 from offering 1, and can mix the two at $p = 0.9$.

What happens in reality? Since the first experiment on the game (Güth, Schmittberger, and Schwarze, 1982), a similar picture has emerged in hundreds of experiments. The most common offer is 40–50% of the pie. About half of the offers below 20% are rejected (Camerer, 2003). People punish offers they find unfair at a cost to themselves. This behavior does not fit the model of a player who "cares only about tokens".

In the interactive Ultimatum Game, the computer plays the responder at three different levels. At each level the smallest offer the responder accepts is different. The proposer's best response each time is to offer the smallest amount the responder accepts. Table 4.1 compares these cases.

Responder in the game	Smallest offer accepted	Proposer's payoff
Theory (tokens only)	1	9
Level 1: lenient	1	9
Level 2: strategic	3	7
Level 3: fairness-minded	5	5

Table 4.1: Responder types in the Ultimatum Game and the proposer's best response (10 tokens).

The "Theory" row of the table corresponds to the (9, 1) equilibrium in which the responder rejects 0; if the responder also accepts 0, the proposer offers 0 and gets 10. Against the fairness-minded responder, the proposer cannot offer less than 5; if they do, they are rejected and get 0. The proposer's payoff falls as the responder's acceptance threshold rises.

The lesson for mechanism design is this: when designing an incentive-compatible mechanism, the player's utility function must be chosen correctly. If the player also cares about fairness, the IR condition is not "$x \geq 0$" but "x reaches the share the player considers fair". A mechanism built on a wrong utility model looks IR on paper but is rejected in practice. The discussion of behavior and ethics (4.6.1) returns to this topic.

4.2.6 Proportional cost sharing, step by step

The free-rider problem (4.1.4) can also be examined with numbers. Three residents, Ayşe, Bora, and Cem, want a shared park. The park costs 60,000 TL. The residents' true values for the park are 30, 25, and 20 thousand TL, respectively. The values sum to 75. Since 75 > 60, building the park is efficient: the park creates more value than it costs.

The municipality applies the following rule. Each resident reports the value they place on the park. If the reports sum to 60 or more, the park is built and the cost is shared in proportion to the reports: resident i pays $60 \cdot b_i / (b_1 + b_2 + b_3)$. If the total is less than 60, the park is not built and no one pays. The rule looks fair: whoever values the park more pays more. The question is whether the rule is incentive compatible.

Case	Reports (A, B, C)	Total	Park built	Payments	Utilities $u = v - payment$
Everyone honest	30, 25, 20	75	Yes	24, 20, 16	6, 5, 4
Cem reports 5	30, 25, 5	60	Yes	30, 25, 5	0, 0, 15
Everyone reports half	15, 12.5, 10	37.5	No	0, 0, 0	0, 0, 0

Table 4.2: Proportional cost sharing. The park costs 60 thousand TL; values, payments, and utilities are in thousands of TL.

Example 4.6

1. **Everyone honest.** The reports are 30, 25, 20; the total is $75 \geq 60$, so the park is built. The payments are $60 \cdot 30/75 = 24$, $60 \cdot 25/75 = 20$, and $60 \cdot 20/75 = 16$; they sum to 60. The utilities are $30 - 24 = 6$, $25 - 20 = 5$, $20 - 16 = 4$. Everyone's utility is at least 0, so IR holds. The total utility is 15, which equals the difference between the sum of values, 75, and the cost, 60.
2. **Cem lies.** While Ayşe and Bora stay honest, Cem reports a value of 5. The total is $30 + 25 + 5 = 60$, so the park is still built. The payments equal the reports: $60 \cdot 30/60 = 30$, 25, and 5. Cem pays 5 but still enjoys the park at a value of 20. His utility is $20 - 5 = 15$. When honest it was 4; the lie gained Cem 11 units. The rule is not IC.
3. **Cem's most profitable lie.** Let Cem's report be c. For the park to be built, we need $55 + c \geq 60$, that is, $c \geq 5$. When the park is built, Cem's payment is $60c/(55 + c)$, which increases with c. Since Cem wants the park built and wants to pay as little as possible, he chooses $c = 5$: the total is exactly 60.

4. **Everyone acts on the same logic.** Each resident, relying on the others' reports, wants to lower their own report. If everyone reports half their value, the total is $15 + 12.5 + 10 = 37.5 < 60$. The park is not built; the total utility falls from 15 to 0.

Free riding is therefore not only a fairness problem but an efficiency problem. A park that everyone wants and whose total value exceeds its cost may not be built because of the incentives the rule creates.

The source of the problem is clear: a resident's report directly affects their own payment even when the outcome does not change. When Cem says 5 instead of 20, the park is still built; Cem's lie does not change the outcome, it only lowers his own payment. The idea for an incentive-compatible rule is this: a player's report should be able to affect their payment *only by changing the chosen outcome*. If the outcome stays the same, the payment should stay the same. This idea is the basis of the VCG mechanism (4.3).

Common mistake

"Everyone gains when honest, so the rule is good." IR holding under honest reporting does not show that the rule is IC. To test IC, we must check whether each player, holding the others fixed, could gain more utility with another report. Under proportional sharing, Cem deviating alone gets 15 instead of 4.

4.2.7 Design criteria and trade-offs

Four basic criteria are used to evaluate a mechanism. The first two concern incentives, the last two the quality of the outcome.

- **Incentive compatibility (IC):** revealing the true information is the best option. It can be required at the dominant-strategy level (DSIC) or at the Bayesian level (BIC).
- **Individual rationality (IR):** participating is no worse than staying out.
- **Efficiency:** under honest reporting, the chosen outcome is the one with the highest social welfare. The **social welfare** of an outcome x is $SW(x) = \Sigma_i\, v_i(x)$, that is, the sum of the values the players place on that outcome.
- **Budget balance:** this concerns the sum of the payments. **Weak budget balance** means the sum of the payments is at least 0: the mechanism needs no money from outside. **Strong budget balance** means the sum of the payments is exactly 0: no outside money is needed and no money is left over.

The symbol Σ means "sum". The expression $\Sigma_i\, v_i(x)$ means that i runs over all players and the values $v_i(x)$ are added up. It is the same as writing $v_1(x) + v_2(x) + v_3(x)$ for three players.



Figure 4.8: Four example mechanisms and four criteria. Each gives up at least one criterion: VCG gives up strong budget balance, proportional sharing and the midpoint rule give up incentive compatibility, and the fixed price gives up efficiency.

Figure 4.8 compares four mechanisms against these criteria. VCG and proportional sharing come from the public project example, and the fixed price and the midpoint from **bilateral trade**. In bilateral trade there is one seller and one buyer. The seller's cost of giving up the good is c, and the buyer's value for the good is v. Both are private information. Trade is efficient only when $v \geq c$: the good should end up with the one who values it more.

Example 4.7

Fixed price. An intermediary announces the price in advance as 50. The buyer says "I buy" or "I don't buy", the seller says "I sell" or "I don't sell"; if both say yes, trade takes place at 50.

1. The buyer's best response is "I buy" if $v \geq 50$. The seller's best response is "I sell" if $c \leq 50$. These responses do not depend on what the other side says: the rule is DSIC.
2. When trade takes place, the buyer gains $v - 50 \geq 0$ and the seller gains $50 - c \geq 0$: IR holds. What the buyer pays goes to the seller: there is strong budget balance.
3. But let $v = 45$ and $c = 40$. Trade is efficient ($45 > 40$), but it does not happen because the buyer is unwilling to pay 50. The rule is not efficient.

Midpoint rule. The buyer reports a value and the seller reports a cost. If the buyer's report is at least the seller's, trade takes place at the price midway between the two reports.

1. Let $v = 60$ and $c = 40$, and both report honestly. The price is $(60 + 40)/2 = 50$. The buyer gains 10 and the seller gains 10. Under honest reporting, trade takes place exactly when it is efficient and no one loses.
2. If the buyer reports a value of 52, trade still takes place, but the price becomes $(52 + 40)/2 = 46$. The buyer's utility is $60 - 46 = 14 > 10$. Understating the value paid off: the rule is not IC.

The trade-off in these examples is no coincidence. Myerson and Satterthwaite (1983) showed that it is a general result.

Theorem 4.3 (Myerson–Satterthwaite)

In bilateral trade where the buyer's value and the seller's cost are independent and drawn from overlapping ranges, no mechanism is simultaneously efficient, (Bayesian) incentive compatible, individually rational, and budget balanced.

The proof of the theorem is beyond the scope of these notes. What matters is the meaning of the result: a designer who wants all four criteria at once is asking for the impossible. Mechanism design is therefore often a problem of trade-offs. The designer decides which criterion to give up and by how much. For example, VCG preserves efficiency and DSIC but runs a budget deficit in bilateral trade: the buyer pays the seller's cost, the seller receives the buyer's value, and someone from outside must cover the difference; a fixed price preserves the budget and incentives but misses some efficient trades.

Common mistake

Reading the Myerson–Satterthwaite theorem as "trade is impossible". The theorem does not say that trade cannot happen; it says only that no rule satisfies all four properties *at once*. The fixed-price rule is used every day and makes many efficient trades possible; it just misses some.

4.2.8 Implementation theory: is truth-telling the only equilibrium?

The revelation principle says that honest reporting is an equilibrium. But the same mechanism may also have other, unwanted equilibria. The rules make a good equilibrium possible; they do not guarantee that the players will meet at that equilibrium.

Requiring that *all* equilibria of a mechanism lead to the desired outcome is a stronger demand. The field that studies such mechanisms is called **implementation theory**. One of its basic results is Maskin's work characterizing the social choice functions that can be implemented in Nash equilibrium (Maskin, 1999). An example shows why the question matters.

Example 4.8

In a second-price (Vickrey) auction let Ayşe's value be 10 and Bora's value 5. Ayşe bids 0 and Bora bids 100.

1. **Outcome.** The highest bid is Bora's; Bora wins and pays the second-highest bid, which is 0. Bora's utility is $5 - 0 = 5$ and Ayşe's is 0.

2. **Can Ayşe deviate?** With any bid below 100, Ayşe loses and gets 0. To win she has to exceed 100; she would then pay Bora's bid of 100 and her utility would be $10 - 100 = -90$. Ayşe's deviation is not profitable.
3. **Can Bora deviate?** Bora is currently winning for free and gets 5. No bid can give him more than 5, since his value for the item is 5 and the payment cannot fall below 0.
4. **Conclusion.** No one wants to deviate alone: the bids (0, 100) form a Nash equilibrium. But the item goes to Bora, who has the lower value; the outcome is not efficient.

This bad equilibrium does not contradict the mechanism being DSIC. The truthful bid is still a dominant strategy for both players. The bad equilibrium is built from **weakly dominated** bids: Bora's bid of 100 is never better than the truthful bid of 5 and in some cases (for example if Ayşe bids 50) is worse. Ayşe's bid of 0 is likewise weakly dominated by the truthful bid of 10.

The lesson is twofold. In DSIC mechanisms a player need not guess the rival; honesty is a safe choice in every case. DSIC designs are therefore more robust than designs that rest only on Nash equilibrium. Even so, DSIC gives no guarantee of a unique equilibrium. If players do not play their dominant strategies, odd equilibria can arise.

4.2.9 Single-parameter settings: Myerson's lemma

In many important problems each player's private information is a single number. In selling a single item, this number is the value placed on the item. When several identical items are sold and each player wants at most one, a single number again suffices. Settings of this kind are called **single-parameter environments**.

In a single-parameter environment, each player has a **unit value** v_i. From the reports b, the mechanism chooses for each player an **allocation** $x_i(b)$ and a payment $p_i(b)$. The allocation says how much "good" the player receives: with a single item it is 0 or 1; if the item is given by lottery, it can be a probability between 0 and 1. The player's utility is $v_i \cdot x_i - p_i$.

Myerson (1981) proved a result that characterizes DSIC mechanisms in single-parameter environments exactly. The result answers two questions: which allocation rules can be used in a DSIC mechanism, and what must the payments then be?

Definition 4.6

An allocation rule x is **monotone** if, for every player i and every report profile b_{-i} of the others, $x_i(z, b_{-i})$ is non-decreasing in z. That is, a player who raises their bid must not receive less of the good.

Theorem 4.4 (Myerson's lemma)

In a single-parameter environment:

1. An allocation rule x can be used in a DSIC mechanism if and only if it is monotone.
2. If x is monotone, there is a unique payment rule that makes the mechanism DSIC subject to $p_i(0, b_{-i}) = 0$:

$$p_i(b) = b_i \cdot x_i(b) - \int_0^{b_i} x_i(z, b_{-i})\, dz$$

The integral sign ($\int$) in the formula may look intimidating, but here it only expresses an area. $\int_0^{b_i} x_i(z, b_{-i})\, dz$ is the area under the curve x_i on the horizontal axis from 0 to b_i. In all the examples in this chapter the curve is a step function. The area under a step curve is found by adding up the areas of rectangles: width × height for each rectangle.

The geometric reading of the formula is as follows. $b_i \cdot x_i(b)$ is the area of the rectangle with width b_i and height $x_i(b)$. When the area under the curve is subtracted from this rectangle, what remains is the area to the left of the curve, between heights 0 and $x(b)$. The payment is this shaded area. The area under the curve, in turn, is the player's utility: under honest reporting, $v_i\, x_i - p_i = \int_0^{v_i} x_i(z)\, dz$.

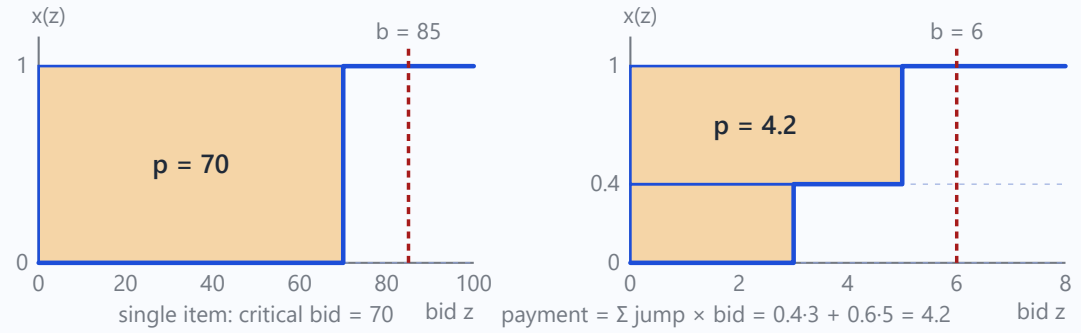


Figure 4.9: The Myerson payment is the shaded area. On the left, a single item: the allocation jumps from 0 to 1 at the bid 70; at the bid $b = 85$ the payment is $p = 70$, the critical bid. On the right, a step allocation: it jumps to 0.4 at the bid 3 and to 1 at 5; at $b = 6$ the payment is $0.4 \cdot 3 + 0.6 \cdot 5 = 4.2$.

Example 4.9

A single item (left of the figure). Let the highest bid of the others be 70. If the player bids above 70, they get the item ($x = 1$); if they bid below, they do not ($x = 0$). The player's bid is $b = 85$.

1. $b \cdot x(b) = 85 \cdot 1 = 85$.
2. Area under the curve: between 0 and 70 the height is 0, so the area is 0; between 70 and 85 the height is 1, so the area is $15 \cdot 1 = 15$. The total is 15.

3. The payment is $85 - 15 = 70$. The utility is $85 - 70 = 15$, which is the area under the curve.

A step allocation (right of the figure). Let the allocation be 0 below a bid of 3, 0.4 between 3 and 5, and 1 from 5 on. The bid is $b = 6$.

1. $b \cdot x(b) = 6 \cdot 1 = 6$.
2. Area under the curve: 0 on [0, 3); $2 \cdot 0.4 = 0.8$ on [3, 5); $1 \cdot 1 = 1$ on [5, 6]. The total is 1.8.
3. The payment is $6 - 1.8 = 4.2$.
4. The same result can be found by a shorter route. For a step allocation, the payment is the sum, over all jumps, of the size of the jump times the bid at which it occurs: $0.4 \cdot 3 + 0.6 \cdot 5 = 1.2 + 3 = 4.2$.

The reason for the step formula is this: each jump in the allocation is an extra "piece of the good" that the player obtains by crossing that threshold. The player pays exactly the threshold for each unit of that piece. By crossing 3 the player gets 0.4 units of the good and pays 3 for each unit; by crossing 5 the player gets 0.6 units more and pays 5 for each unit.

The critical bid. If the allocation can only be 0 or 1, there is a single jump. The payment is then the lowest bid the player needs in order to win. This bid is called the **critical bid**. In selling a single item, the critical bid is the highest bid of the others. Myerson's lemma therefore says: for a single item, the only DSIC payment rule that gives the item to the highest bidder is the Vickrey rule, in which the winner pays the second-highest bid (losers pay 0).

What if the payment is not the critical bid? Let the threshold be 50 and the winner's payment a fixed 20. A player with value 35 who bids honestly stays below the threshold and gets 0. If the player bids above 50, they get the item and pay 20; the utility is $35 - 20 = 15$. Overstating paid off. In general, every player whose value lies between the payment and the threshold (here between 20 and 50) wants to overstate; IC fails. Had the payment been larger than the threshold, winners whose value lies between the threshold and the payment would lose money and would want to lower their bids below the threshold.

Why is monotonicity necessary? A short argument shows it. The honest report of a player with value y must be no worse than reporting z; the honest report of a player with value z must likewise be no worse than reporting y:

$$y \cdot x(y) - p(y) \geq y \cdot x(z) - p(z) \text{ and } z \cdot x(z) - p(z) \geq z \cdot x(y) - p(y)$$

Adding the two inequalities side by side cancels the payments and leaves $y \cdot x(y) + z \cdot x(z) \geq y \cdot x(z) + z \cdot x(y)$. This can be written as $(z - y)(x(z) - x(y)) \geq 0$. If $z > y$, we must have $x(z) \geq x(y)$: a higher bid cannot receive a smaller allocation. So DSIC forces monotonicity. The other direction of the lemma, that every monotone allocation is DSIC with the payment in the formula, is proved step by step in Roughgarden's (2016) lecture notes.

Common mistake

Mistaking the Myerson payment for the area *under* the curve. The area under the curve is the player's utility. The payment is what remains when this area is subtracted from the rectangle $b \cdot x(b)$: the area to the left of the curve. In the single-item example the area under the curve is 15 (the utility) and the area to the left is 70 (the payment).

4.3 The VCG mechanism

The failure of proportional cost sharing (4.2.6) left us with an idea for a solution: a player's report should affect their payment only by changing the chosen outcome. The Vickrey–Clarke–Groves (VCG) mechanism turns this idea into a general rule. The mechanism combines William Vickrey's (1961) single-item auction, Edward Clarke's (1971) work on public goods, and Theodore Groves's (1973) general framework.

4.3.1 Efficient choice and the Clarke pivot payment

VCG works in two steps. In the first step the outcome is chosen; in the second, the payments are computed. The choice of outcome is efficient: the outcome with the highest social welfare under the reported values is selected. The payment equals the harm that each player's presence does to the others.

Definition 4.7

The **VCG mechanism** (Vickrey–Clarke–Groves mechanism) asks each player for the value placed on every possible outcome; $b_i(x)$ is the value that player i reports for outcome x. The mechanism chooses the outcome

$$x^* = \operatorname{argmax}_x \Sigma_i \, b_i(x)$$

and charges each player according to the **Clarke pivot rule**:

$$p_i = \max_x \Sigma_{j \neq i} \, b_j(x) - \Sigma_{j \neq i} \, b_j(x^*)$$

It helps to read the two formulas in words. *argmax* returns the *option* that gives the largest value, while *max* returns the largest *value*. x^* is the outcome that maximizes the sum of the values reported by all players. In the payment formula, $\Sigma_{j \neq i}$ means "sum over all players other than i". The formula has two parts:

- **First term:** if *i* were absent, the other players would choose the best outcome among themselves and would get this much welfare from it. In short, "the others' best welfare without *i*".
- **Second term:** the welfare the others get at the outcome *x** chosen when *i* is present. In short, "the others' welfare with *i*".

The difference is the **externality** that *i*'s participation imposes on the others. An externality is the harm (or benefit) that one person's decision causes to others without that person paying for it. A factory polluting the air is a negative externality. VCG makes each player pay for the negative externality they create. If a player does not change the outcome in their own favor, the first and second terms are equal and the player pays nothing. A player who changes the outcome, the "pivot" player, pays the cost of that change to the others. This is where the rule gets its name.

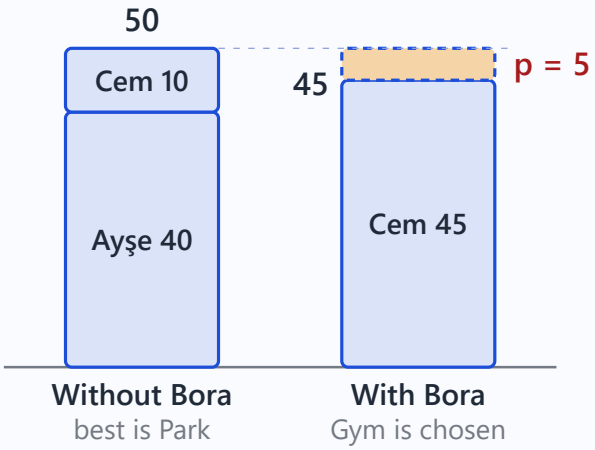


Figure 4.10: Bora's externality in the public project example. Without Bora, the best project for the others is the Park, and their welfare is *40 + 10 = 50*. With Bora, the Gym is chosen and the others' welfare is *0 + 45 = 45*. The difference is *p = 5*, Bora's payment.

Figure 4.10 shows this idea with a pair of columns. The left column shows the welfare the others would get without Bora, and the right column the welfare they get with Bora. Bora's presence shifts the choice from the Park to the Gym and lowers the others' welfare from 50 to 45. Bora pays for this loss of 5 units. All the numbers of this example are in 4.3.2.

Why is VCG DSIC? The first term of the payment formula depends only on the others' reports; it does not change whatever *i* reports. Call this term h_i. Let the others report b_{-i} and let *i*'s report lead to outcome *x*. Player *i*'s true utility is then:

$$u_i = v_i(x) - p_i = v_i(x) + \Sigma_{j \neq i}\, b_j(x) - h_i$$

Theorem 4.5 (Incentive compatibility of VCG)

In the VCG mechanism, honest reporting is a dominant strategy for every player. Clarke pivot payments are never negative; if values are also non-negative, the mechanism is individually rational.

Proof. **DSIC.** The last term of the utility above, h_i, does not depend on i's report. To maximize utility, i therefore wants the outcome x that maximizes $v_i(x) + \Sigma_{j \neq i}\, b_j(x)$. This sum is i's *true* value plus the sum of the others' reports. The mechanism chooses the outcome that maximizes $b_i(x) + \Sigma_{j \neq i}\, b_j(x)$. If i reports honestly ($b_i = v_i$), the two sums are identical and the mechanism chooses exactly the outcome i most wants. A false report can lead the mechanism either to the same outcome or to an outcome that makes the sum i wants smaller. No lie can do better than honesty, and this holds whatever the others report.

Payments are non-negative. The first term is the *largest* welfare the others can get over all outcomes. The second term is the welfare the others get at one particular outcome (x^*). A maximum cannot be smaller than any particular value; hence $p_i \geq 0$.

IR. Under honest reporting, $u_i = \Sigma_j\, v_j(x^*) - max_x\, \Sigma_{j \neq i}\, v_j(x)$. Let x' be the others' best outcome without i. Since x^* maximizes total welfare, $\Sigma_j\, v_j(x^*) \geq \Sigma_j\, v_j(x') = v_i(x') + \Sigma_{j \neq i}\, v_j(x')$. Since values are non-negative, $v_i(x') \geq 0$, and so $u_i \geq 0$. □

The main idea of the proof can be put in one sentence: in VCG, a player's utility is social welfare minus a term that is independent of that player. Every player therefore wants social welfare to be as large as possible, and the mechanism does exactly that. The player's interest and society's interest are aligned.

4.3.2 Choosing a public project, step by step

Three residents, Ayşe, Bora, and Cem, will choose one of three projects: the Park, the Library, and the Gym. Only one project will be built. Each resident reports the value (in thousands of TL) they place on each project; everyone is assumed to report honestly.

	Park	Library	Gym	Payment p_i	Utility u_i
Ayşe	40	5	0	0	0
Bora	0	30	20	5	15
Cem	10	20	45	20	25
(a) Σ everyone	50	55	**65 ✓**	25	40
(b) Σ without Ayşe	10	50	65		
(b) Σ without Bora	50	25	45		
(b) Σ without Cem	40	35	20		

Table 4.3: Choosing a public project with VCG. Values are in thousands of TL; ✓ marks the chosen project. The last two columns are filled in once the computation is done.

Example 4.10

The computation follows a four-step pattern: (a) the best choice with everyone, (b) the best choice without i, (c) the others' welfare at the project chosen when i is present, (d) payment = (b) − (c).

1. **(a) The best choice with everyone.** Add the three values for each project. Park: $40 + 0 + 10 = 50$. Library: $5 + 30 + 20 = 55$. Gym: $0 + 20 + 45 = 65$. The largest total is for the Gym: x^* = Gym, with social welfare 65.
2. **Ayşe (b).** Without Ayşe, add only Bora's and Cem's values: Park $0 + 10 = 10$, Library $30 + 20 = 50$, Gym $20 + 45 = 65$. The best is the Gym; the others' welfare is 65.
3. **Ayşe (c).** With Ayşe present, the chosen project is the Gym. There the others' welfare is Bora 20 + Cem 45 = 65.
4. **Ayşe (d).** $p_{Ayşe} = 65 - 65 = 0$. The Gym is chosen whether or not Ayşe is present; Ayşe does not harm the others.
5. **Bora (b).** Without Bora, the totals for Ayşe and Cem: Park $40 + 10 = 50$, Library $5 + 20 = 25$, Gym $0 + 45 = 45$. The best is the Park; the others' welfare is 50.
6. **Bora (c) and (d).** With Bora present, the Gym is chosen and the others' welfare is Ayşe 0 + Cem 45 = 45. $p_{Bora} = 50 - 45 = 5$. Without Bora the Park would be chosen and the others would get 50; because of Bora the Gym is chosen and the others get 45. The harm Bora does to the others is 5.
7. **Cem (b).** Without Cem, the totals for Ayşe and Bora: Park $40 + 0 = 40$, Library $5 + 30 = 35$, Gym $0 + 20 = 20$. The best is the Park; the others' welfare is 40.
8. **Cem (c) and (d).** With Cem present, the Gym is chosen and the others' welfare is Ayşe 0 + Bora 20 = 20. $p_{Cem} = 40 - 20 = 20$. The harm Cem does to the others is 20.
9. **Utilities.** Utility = value of the chosen project − payment. Ayşe $0 - 0 = 0$, Bora $20 - 5 = 15$, Cem $45 - 20 = 25$. Everyone's utility is at least 0 (IR). The total payment collected is $0 + 5 + 20 = 25$.

Interpreting the results reinforces the logic of the mechanism. Ayşe wanted the Park most, but her preference did not change the outcome; Ayşe pays nothing. Bora and Cem are each pivotal on their own: without either of them the Park would be chosen. So both pay. Cem's payment is larger because his presence changes the choice in a more costly way: without Cem, Ayşe and Bora would get 40 from the Park, whereas with the Gym they get only 20.

The 25 units collected are not redistributed to anyone. This means VCG produces a **budget surplus**. If the money were returned to the residents, the incentives could break, because the shares returned would depend on the players' reports. Since payments are non-negative, the mechanism needs no outside money: weak budget balance holds, strong budget balance does not.

Common mistake

Taking "the others' best welfare with *i* present" in step (c). Step (c) is always computed at x^*, the project chosen with everyone; not at the project the others like best at that moment. For Bora, (c) is the 45 at the Gym, not the 50 at the Park. Another common mistake is to leave *i*'s own value in the sum in step (b).

4.3.3 Code trace: vcg()

The whole VCG computation can be written as a short Python function. The function takes two inputs. `v` is a dictionary: `v[i][a]` is the value player `i` reports for outcome `a`. `A` is the list of possible outcomes. The function returns the chosen outcome and the payments.

```
def vcg(v, A):
  N = list(v)            # players
  def w(a, S):           # welfare of S at a
    return sum(v[i][a] for i in S)
  x = max(A, key=lambda a: w(a, N))
  p = {}
  for i in N:
    D = [j for j in N if j != i]
    w_out = max(w(a, D) for a in A)  # i absent
    w_in = w(x, D)                   # i present
    p[i] = w_out - w_in
  return x, p
```

The code works line by line as follows:

- `N = list(v)` extracts the list of players from the dictionary keys: `['Ayşe', 'Bora', 'Cem']`.
- The inner function `w(a, S)` is the sum of the values that the players in the set `S` place on outcome `a`. `w(a, N)` is everyone's welfare, and `w(a, D)` is the welfare of the set `D` alone.
- `x = max(A, key=lambda a: w(a, N))` picks, among the outcomes in `A`, the one that maximizes `w(a, N)`. This is the Python counterpart of the argmax operation.
- `p = {}` is an empty payment dictionary.
- In the loop, for each player `i`, `D` is the list of players other than `i`. `w_out` is the largest welfare of `D` over all outcomes: the first term of the payment formula. `w_in` is the welfare of `D` at the chosen outcome `x`: the second term. The payment is `w_out − w_in`.

In the public project example the input is `v = {'Ayşe': {'P': 40, 'L': 5, 'G': 0}, 'Bora': {'P': 0, 'L': 30, 'G': 20}, 'Cem': {'P': 10, 'L': 20, 'G': 45}}` and `A = ['P', 'L', 'G']`. Table 4.4 shows each step of the loop.

Step	w(·, N) or w(·, D): P, L, G	w_out	w_in	p
choice of x	w(·, N) = 50, 55, 65 → x = G			{}
i = Ayşe	D = [Bora, Cem]: 10, 50, 65	65	65	{Ayşe: 0}
i = Bora	D = [Ayşe, Cem]: 50, 25, 45	50	45	{..., Bora: 5}
i = Cem	D = [Ayşe, Bora]: 40, 35, 20	40	20	{..., Cem: 20}
return	(G, {Ayşe: 0, Bora: 5, Cem: 20}); total 25			

Table 4.4: Trace of `vcg(v, A)`. P = Park, L = Library, G = Gym.

The trace matches the hand computation exactly. The simplicity of the code comes from VCG being a general mechanism: whatever the outcomes are (projects, allocations of items, routes), the same function works. The only requirement is that every player can report a value for every outcome.

Two details about the code matter. First, on ties `max` picks the outcome that comes first in the list; this fixed tie-breaking rule does not affect DSIC. Second, for $|A|$ outcomes and n players, the code does $|A| \cdot n$ additions to find x, and repeats about the same work for each player's payment. The total work is $O(n^2 \cdot |A|)$. This is very fast when the set of outcomes is small. But when the outcomes are combinatorially numerous, for example when they are all possible ways of distributing 30 items among the players, $|A|$ grows astronomically. This problem is taken up in 4.3.6.

4.3.4 Can lying pay off under VCG?

By the theorem, lying does not pay off under VCG. Testing a few lies directly in the public project example makes the proof concrete. In each row only one player lies; the others are honest. The last column is the liar's utility according to their *true* value.

Case	Report (P, L, G)	Chosen	Payment	True utility
Cem honest	10, 20, 45	Gym	20	45 − 20 = 25
Cem overstates	10, 20, 100	Gym	20	45 − 20 = 25
Cem understates	10, 20, 25	Library	5	20 − 5 = 15
Bora overstates	0, 60, 20	Library	25	30 − 25 = 5

Table 4.5: Single-player lies under VCG. The report (P, L, G) gives the values placed on the Park, the Library, and the Gym, respectively.

Example 4.11

1. **Cem reports 100 for the Gym.** The totals are 50, 55, 120; the Gym is chosen again. Cem's payment is (b) − (c) = 40 − 20 = 20: both terms depend only on Ayşe's and Bora's reports and do not change. Cem's utility is still *45 − 20 = 25*. Overstating changed nothing.
2. **Cem reports 25 for the Gym.** The totals are Park 50, Library 55, Gym *0 + 20 + 25 = 45*; the Library is chosen. Cem's payment: without Cem the Park would be chosen and the others would get 40; at the Library the others get *5 + 30 = 35*. The payment is *40 − 35 = 5*. Cem's

true value for the Library is 20; his utility is $20 - 5 = 15 < 25$. Understating cost Cem the project he wanted most.

3. **Bora reports 60 for the Library.** The totals are Park 50, Library $5 + 60 + 20 = 85$, Gym 65; the Library is chosen. Bora's payment: without Bora the Park would be chosen and the others would get 50; at the Library the others get $5 + 20 = 25$. The payment is $50 - 25 = 25$. Bora's true value for the Library is 30; his utility is $30 - 25 = 5 < 15$.

Bora's lie is interesting. Bora really did prefer the Library to the Gym (30 > 20). By lying he got his favorite project built. But the price rose by exactly the harm he did to the others: the Library is 20 units worse for the others than the Gym. Bora's gain (10 more units of value at the Library) does not cover this price. This is exactly how VCG works: changing the outcome is allowed, but the cost of the change to others is paid. If the change is not good for society, it is not good for the liar either.

These four cases are only examples. A wider experiment can also be run on a computer: let one player report each of 0, 10, ..., 100 for each project while the other two stay honest. Under none of these reports can any player obtain higher utility than by reporting honestly. This is a numerical check of Theorem 4.5.

4.3.5 VCG for a single item: the Vickrey auction

VCG is a general mechanism; selling a single item is a special case. In this special case, VCG can be shown to reduce to exactly the second-price (Vickrey) auction. To see this, the same `vcg()` function is applied to a single item.

With a single item, an outcome is the answer to "who gets the item?". The outcomes are the players themselves: `A = ['Ayşe', 'Bora', 'Cem']`. Player `i`'s value for outcome `a` is the value placed on the item if `a = i` (the item goes to that player) and 0 otherwise. Let the values be 70, 55, and 85. For example, `v['Ayşe'] = {'Ayşe': 70, 'Bora': 0, 'Cem': 0}`.

Step	a = Ayşe	a = Bora	a = Cem	w_out	w_in	p
w(a, N)	70	55	**85 ✓**			x = Cem
i = Ayşe, w(a, D)	0	55	85	85	85	0
i = Bora, w(a, D)	70	0	85	85	85	0
i = Cem, w(a, D)	70	55	0	70	0	70

Table 4.6: Trace of `vcg()` for a single item. The values are Ayşe 70, Bora 55, Cem 85. The columns are the outcomes "who gets the item?".

Example 4.12

1. **Choosing the outcome.** The welfare of each outcome is just the value of the person who gets the item, since the others get 0: w(Ayşe, N) = 70, w(Bora, N) = 55, w(Cem, N) = 85. The largest is Cem: the item goes to the player with the highest value.

2. **i = Ayşe.** D = [Bora, Cem]. Without Ayşe the item would go to Cem: `w_out` = 85. The item is already with Cem; D's welfare there is `w_in` = 85. p[Ayşe] = 85 − 85 = 0: the loser pays nothing.
3. **i = Bora.** D = [Ayşe, Cem]. Without Bora the item would still go to Cem: `w_out` = 85; `w_in` = 85. p[Bora] = 0.
4. **i = Cem.** D = [Ayşe, Bora]. Without Cem the item would go to Ayşe: `w_out` = 70. Since the item is with Cem, D's welfare is `w_in` = 0. p[Cem] = 70 − 0 = 70.
5. **Result.** Cem wins and pays 70, the second-highest value. Losers pay nothing. This is exactly the Vickrey (second-price) auction.

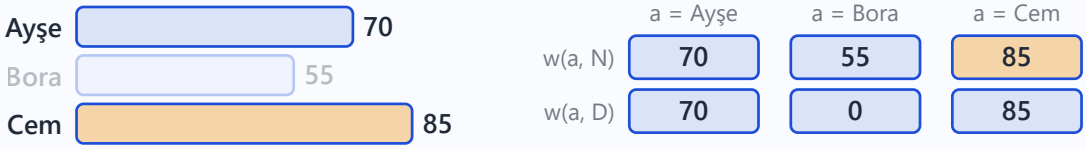


Figure 4.11: One step of the VCG trace for a single item. For i = Bora, Bora's row is dimmed; for D = {Ayşe, Cem}, $w(a, D)$ = 70, 0, 85. Without Bora the item would still go to Cem; Bora pays nothing.

The general reason for the result is clear. Without the winner, the item would go to the player with the second-highest value, and the others' welfare would be that second-highest value. With the winner present, the others get nothing. The difference is the second-highest value. If a losing player were absent, the item would still go to the same person; the loser's externality is zero. This result is also consistent with Myerson's lemma (4.2.9): for a single item, the DSIC payment is the critical bid, and the critical bid is the second-highest bid.

4.3.6 Properties and limits of VCG

VCG is a very attractive mechanism in theory. Its strengths are:

- **DSIC:** honest reporting is a dominant strategy; players need not guess one another.
- **Efficiency:** the outcome with the highest social welfare is chosen.
- **IR and non-negative payments:** with the Clarke pivot and non-negative values, no participant loses out and the mechanism never pays anyone.
- **Generality:** for a single item it reduces to the Vickrey auction, and for several identical items to the rule "pay the highest value among the losers".

Nevertheless, VCG is rarely used in practice. There are several reasons:

- **It is not strongly budget balanced.** In the public project example 25 units are left over. The mechanism runs no deficit (weak budget balance), but it cannot redistribute the surplus to the players. In public good examples this money is as if "burned".

- **Computation can be hard.** Computing the *argmax* can be NP-hard in combinatorial problems with many outcomes. For example, finding the best allocation in an auction in which many items are distributed among bidders who each want different bundles of items belongs to the class of hardest known optimization problems. If the computation is approximate rather than exact, VCG's incentive properties can fail.
- **Collusion and false identities.** If players collude secretly, or if one player takes part under several false identities, lying can become profitable. VCG's incentive guarantee covers only individual deviations.
- **Revenue can be low.** Because payments equal externalities, the seller's revenue can be very low, even zero, when there is little competition.
- **Privacy.** Players must reveal the value they place on every outcome. A company may not want to reveal to its rivals and to the seller the highest price it is willing to pay for a product.

Example 4.13

In a single-item Vickrey auction let the values be Ayşe 70, Bora 55, Cem 85. With honest bids Cem wins and pays 70.

1. Ayşe and Cem secretly agree: Ayşe bids 0 and Cem stays honest.
2. The bids are 0, 55, 85. Cem still wins, but the second-highest bid is now Bora's 55. Cem pays 55.
3. Cem's utility rises from $85 - 70 = 15$ to $85 - 55 = 30$. Ayşe got 0 when honest too; she loses nothing from the agreement. The 15 units gained can be shared between them.

If Ayşe alone had lowered her bid, her own utility would not have changed (0). The DSIC definition therefore does not rule out this deviation. But two players deviating together gain as a group. The seller's revenue falls from 70 to 55.

Because of these limits, simpler rules are preferred over VCG in real applications. For example, sponsored search ads mostly use the **generalized second price** (GSP) rule: ads are ranked by their bids and each advertiser pays the bid of the one directly below. GSP is not DSIC, but it is simple and advantageous in terms of revenue. VCG keeps its importance as a theoretical reference point: it shows whether an efficient and incentive-compatible outcome is possible in a problem, and at what cost.

4.4 Mechanisms without money

In the earlier mechanisms, incentives were corrected with payments. VCG made honesty profitable by making each player pay for the externality they create. But in many important problems using money is either forbidden or inappropriate. The main mechanisms that work without money are: Gale–Shapley for stable matching, the median rule for facility location, serial dictatorship and the Boston mechanism for school placement, and TTC for room swaps.

4.4.1 Design without money

In school placement, students are not asked for money to enter a school. In dorm room swaps, students do not pay one another. In an election, votes cannot be bought. In organ transplantation, buying and selling organs is prohibited. In these settings the designer has only one tool left: the outcome rule. Incentives cannot be corrected with payments; the rule must be chosen so that telling the truth does not make the outcome worse for the player.

The term used for incentive compatibility without money is **strategy-proofness**. If a rule is strategy-proof, no player, whatever the others report, can obtain a better outcome by reporting something other than their true preferences. This is the counterpart of DSIC in a setting without money.

For general preferences there is bad news. Gibbard (1973) and Satterthwaite (1975), independently of each other, showed the following:

Theorem 4.6 (Gibbard–Satterthwaite)

In a setting with at least three alternatives, in which every alternative can be the outcome at some preference profile and players can rank the alternatives in any order, every strategy-proof rule is a dictatorship: the outcome is always the favorite alternative of the same single person.

A dictatorship is of course not an acceptable rule. The theorem says that, for general preferences, there is no reasonable voting rule that guarantees honesty. Mechanism design without money therefore proceeds by relaxing one of the theorem's assumptions. There are two ways out:

- **Use the structure of preferences.** If preferences cannot be arbitrary rankings, the theorem does not apply. For example, if the alternatives are arranged on a line and each player has a single favorite point (single-peaked preferences), the median rule is strategy-proof (4.4.7).
- **Change the goal.** Instead of looking for the "best" alternative, we can look for another criterion. In two-sided markets the goal is stability (Gale–Shapley). When everyone holds a good, the goal is trade (TTC).

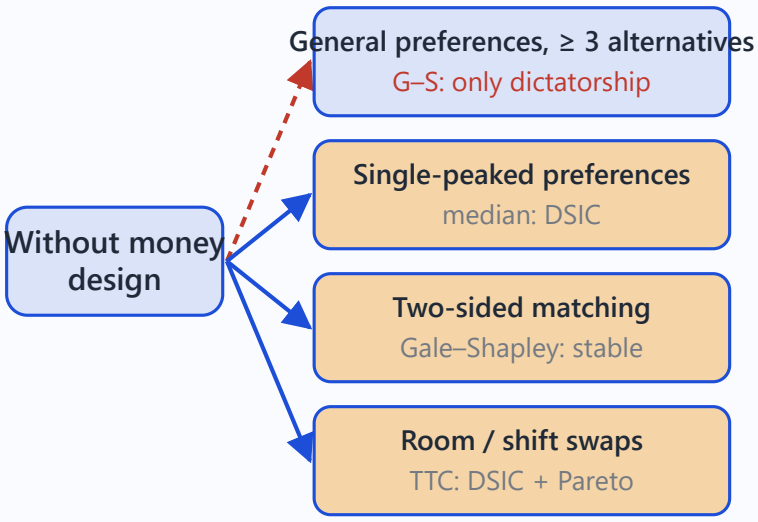


Figure 4.12: Routes in design without money. For general preferences and at least three alternatives, Gibbard–Satterthwaite allows only dictatorship among rules in which every alternative can win. With single-peaked preferences, the median is DSIC; in two-sided matching, Gale–Shapley is stable; in room or shift swaps, TTC is DSIC and Pareto optimal.

4.4.2 The stable matching problem

A department has three students (A, B, C) and three capstone projects (X, Y, Z). Each project takes one student. Each student ranks the projects, and each project's supervisor ranks the students, in order of preference. The preference orders are given in Table 4.7. The notation "$X \succ Y \succ Z$" means "prefers X to Y and Y to Z".

Student	Preference	Project	Preference
A	X ≻ Y ≻ Z	**X**	B ≻ A ≻ C
B	X ≻ Z ≻ Y	**Y**	A ≻ C ≻ B
C	Y ≻ X ≻ Z	**Z**	A ≻ B ≻ C

Table 4.7: Preference orders of the students and the projects.

A **matching** is a list that assigns each student to a different project. For three students and three projects there are $3! = 3 \cdot 2 \cdot 1 = 6$ different matchings. The question of which of them is "good" is answered by asking whether a matching would fall apart.

Definition 4.8

In a matching, a student–project pair who prefer each other to their current partners is called a **blocking pair**. A matching with no blocking pair is called a **stable matching**.

A blocking pair can undermine the matching. Both the student and the supervisor are unhappy with the current situation and want each other; they can set the rule aside and make a deal among themselves. In a stable matching, no one wants to break the arrangement and walk away. For a pair to be blocking, two conditions must hold together: the student must prefer the project to their current project *and* the project must prefer the student to its current student. If only one of them wants it, the pair is not blocking.

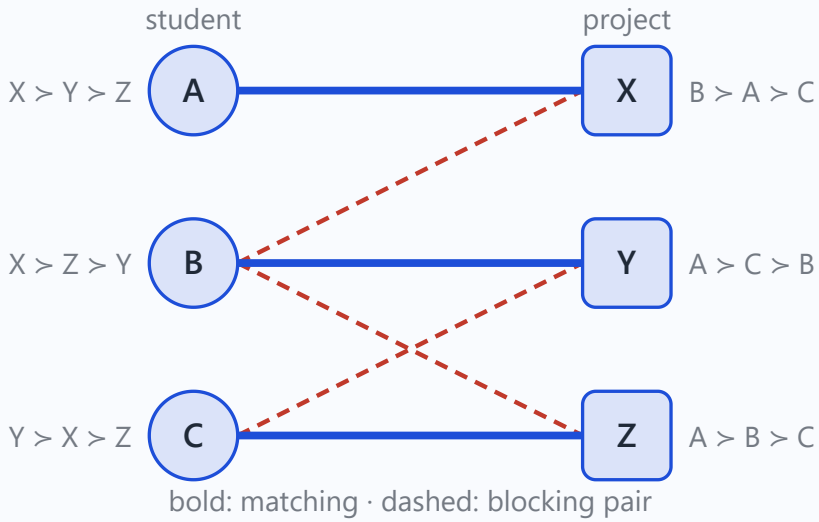


Figure 4.13: An unstable matching. The bold lines show the matching {A–X, B–Y, C–Z}; the red dashed lines show three blocking pairs: (B, X), (B, Z), and (C, Y).

Example 4.14

Is the matching {A–X, B–Y, C–Z} stable? Of the nine student–project pairs, the six that are not in the current matching are checked.

1. **A.** A's first choice is X, and A already has X. A prefers no project to X; there is no blocking pair involving A.
2. **(B, X).** B prefers X to the current project Y (X ≻ Z ≻ Y). X prefers B to its current student A (B ≻ A ≻ C). Blocking.
3. **(B, Z).** B prefers Z to Y. Z prefers B to its current student C (A ≻ B ≻ C). Blocking.
4. **(C, Y).** C prefers Y to the current project Z (Y ≻ X ≻ Z). Y prefers C to its current student B (A ≻ C ≻ B). Blocking.
5. **(C, X).** C prefers X to Z, but X prefers its current student A to C. Not blocking.

The matching has three blocking pairs; it is not stable. Checking all nine pairs turns up no other blocking pair.

Does a stable matching always exist? If so, how is one found? Checking all six matchings is possible in a small example. But for n students there are $n!$ matchings; for 20 students this number exceeds $2 \cdot 10^{18}$. A cleverer method is needed. Gale and Shapley (1962) gave an algorithm that answers both questions at once.

4.4.3 Gale–Shapley: deferred acceptance

In the **Gale–Shapley algorithm**, one side makes proposals and the other side evaluates them. Here the students propose and the projects accept or reject. The algorithm proceeds in rounds:

1. Each free student proposes to their best project that has not yet rejected them.
2. Each project *tentatively* keeps the best of the incoming proposals and the candidate it already holds, and rejects the others.
3. Rejected students become free. When no free student remains, the algorithm stops and the tentative matches become final.

The algorithm's other name is **deferred acceptance**. The name comes from the word "tentatively" in the second step. When a project accepts a proposal it makes no firm commitment; if a better candidate comes along it can drop the earlier one. Firm acceptance is deferred to the end of the algorithm. This detail is the key to why the algorithm produces a stable result.

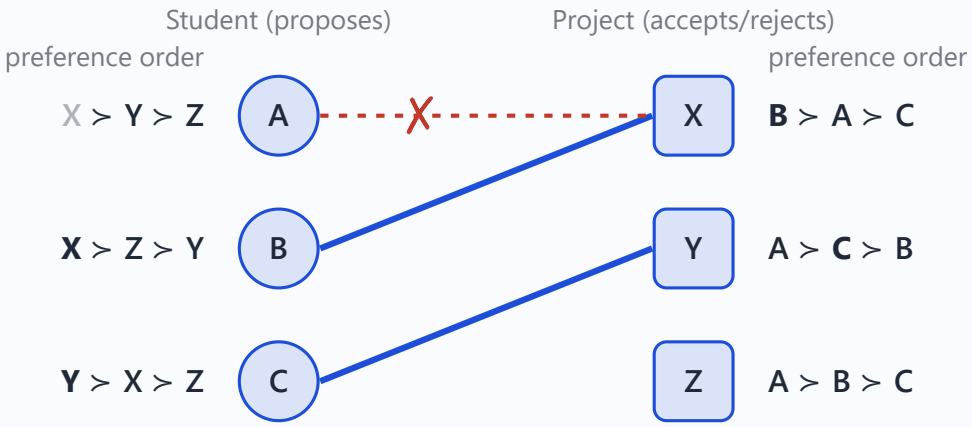


Figure 4.14: Gale–Shapley, round 1 decision. A and B both propose to X; X tentatively keeps B, ranked first, and rejects A (red cross). C's proposal to Y is tentatively accepted. Gray letters are projects to which the student can no longer propose.

Example 4.15

With the preferences in Table 4.7, the algorithm runs as follows.

1. **Start.** Everyone is free and the projects are empty.
2. **Round 1 proposals.** Each student proposes to their first choice: A→X, B→X, C→Y.

3. **Round 1 decision.** X receives two proposals. X's order is B ≻ A ≻ C; B is kept and A is rejected. Only C proposes to Y; C is tentatively accepted.
4. **Round 2 proposals.** Only A is free. X rejected A; the next choice is Y: A→Y.
5. **Round 2 decision.** Y holds C, and now A arrives. Y's order is A ≻ C ≻ B; A is better. Y keeps A and rejects C. C, who had been "accepted" earlier, is now free: the acceptance was tentative.
6. **Round 3 proposals.** Y rejected C; the next choice is X: C→X.
7. **Round 3 decision.** X holds B. X prefers B to C; C is rejected.
8. **Round 4.** C's next choice is Z: C→Z. Z is empty; C is accepted.
9. **End.** Everyone is matched and the algorithm stops. Result: A–Y, B–X, C–Z.

Stability check. B has the first choice X; no complaint. A is with Y and prefers only X to Y, but X prefers B to A. C is with Z and prefers X and Y to Z, but X prefers B to C, and Y prefers A to C. There is no blocking pair: the matching is stable.

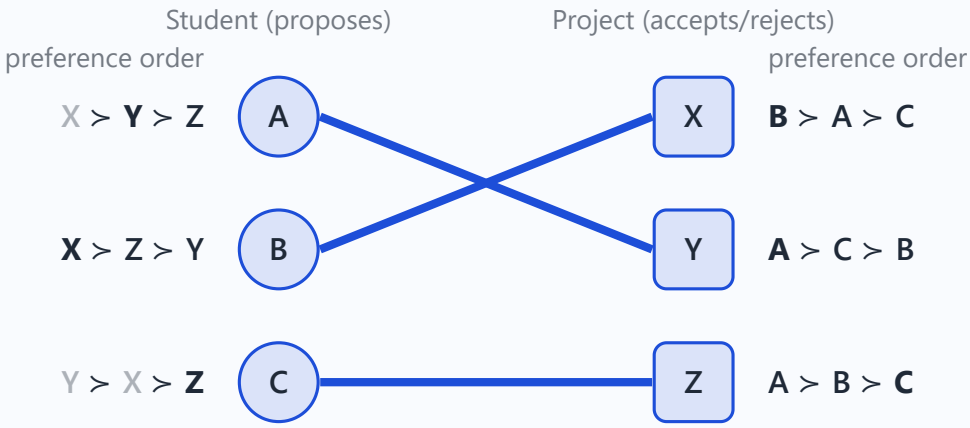


Figure 4.15: The result of Gale–Shapley: A–Y, B–X, C–Z. Bold letters show each student's and project's partner; there is no blocking pair.

Stability does not mean "everyone gets their favorite". C was left with the last choice Z, because X and Y, which C wanted more, are matched to students they like better than C.

4.4.4 Code trace: gale_shapley()

The Python version of the algorithm works with a queue instead of rounds. At each step one free student leaves the queue and makes a proposal. As input, the dictionary `stu` holds each student's preference list (best to worst) and the dictionary `proj` holds the rank each project gives to the students: `proj[p][s]` is the rank that project `p` gives to student `s`, where 0 means best. For example, `proj['X'] = {'B': 0, 'A': 1, 'C': 2}`.

```
def gale_shapley(stu, proj):
  free, match = list(stu), {}
  nxt = {s: 0 for s in stu}
  while free:
    s = free.pop(0)
    p = stu[s][nxt[s]]        # next choice
    nxt[s] += 1
    r = match.get(p)              # p's holder
    if r is None or proj[p][s] < proj[p][r]:
      match[p] = s                # accept
      if r is not None: free.append(r)
    else: free.append(s)    # reject
  return match
```

The parts of the code are:

- `free` is the queue of free students; at the start everyone is in it. The dictionary `match` stores the student each project is tentatively holding.
- `nxt[s]` is the index of the next project in student `s`'s preference list. It starts at 0, the most preferred project.
- In the loop, the student `s` at the head of the queue is removed, proposes to the next choice `p`, and `nxt[s]` is incremented. A student therefore never proposes to the same project twice.
- `r` is the candidate `p` currently holds; it is `None` if the project is empty.
- If the project is empty, or if `p` prefers `s` to `r` (the rank number is smaller), `s` is accepted. If there was an old candidate `r`, it returns to the queue.
- Otherwise `s` is rejected and goes back to the queue.

Proposal	s → p	r (p's holder)	Decision	match (X, Y, Z)	free (after)
1	A → X	None	accept	A, —, —	[B, C]
2	B → X	A	accept B (0 < 1), A back to queue	B, —, —	[C, A]
3	C → Y	None	accept	B, C, —	[A]
4	A → Y	C	accept A (0 < 1), C back to queue	B, A, —	[C]
5	C → X	B	reject (B's rank 0, C's 2)	B, A, —	[C]
6	C → Z	None	accept	B, A, C	[]

Table 4.8: Trace of `gale_shapley(stu, proj)`. Each row is one proposal.

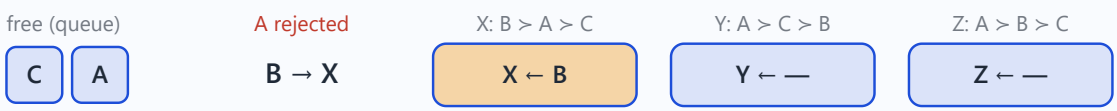


Figure 4.16: The state after the 2nd proposal of the trace. B proposed to X; since B is better for X, B was accepted, and A was rejected and returned to the queue. The queue holds C and A; X ← B, and Y and Z are

empty.

When the queue is empty the algorithm ends after 6 proposals. The result is `match = {'X': 'B', 'Y': 'A', 'Z': 'C'}`, which is the same as the matching A–Y, B–X, C–Z found with simultaneous rounds. Although the proposals are processed in a different order (in the rounds A and B propose at the same time, while in the code they propose one after the other), the result does not change. The result of Gale–Shapley does not depend on the order in which proposals are processed; this is a well-known property of the algorithm.

How many proposals? Each student proposes to each project at most once, because `nxt[s]` only increases. For n students and n projects the total number of proposals is at most $n \cdot n = n^2$. If each proposal is processed in constant time, the running time of the algorithm is $O(n^2)$. (To keep the code short it uses `list.pop(0)`; this operation shifts the list and so is not constant time. For $O(n^2)$, use `collections.deque` and `popleft()`.) For an example with 20 students, at most 400 proposals suffice; this is incomparably faster than checking the $20!$ matchings one by one.

4.4.5 Properties of Gale–Shapley

Theorem 4.7 (Gale and Shapley, 1962)

The Gale–Shapley algorithm terminates on every input and returns a stable matching. Hence, when the numbers of students and projects are equal, a stable matching always exists.

Proof. **Termination.** Each proposal uses up one student–project pair, and there are at most n^2 pairs. The algorithm therefore stops after at most n^2 proposals. It is also impossible for a student to be rejected by every project: once a project has received a proposal, it always holds someone; if a student had been rejected by all projects, all projects would be full, but since the number of projects equals the number of students, this student would then have to be in a project as well.

Stability. Suppose that at the end student s prefers project p to their own partner. Because s proposes following their preference order, s proposed to p before reaching their partner and was rejected by p. A project rejects a student only when it holds a better candidate. The candidate a project holds only improves over time. So at the end p's partner is better than s in p's eyes, and p does not want s. (s, p) cannot be a blocking pair. Since this holds for every pair, the matching is stable. □

The algorithm's other important properties are:

- **Best for the proposing side.** A problem may have more than one stable matching. In Gale–Shapley every proposing player gets the best partner they can get in any stable matching. The receiving side, in turn, gets the worst partner it can get in any stable matching. Which side proposes is therefore an important design decision.

- **DSIC for the proposers.** For the proposing side, reporting the true preference order is a dominant strategy (Roth, 1982). The receiving side can sometimes gain by lying (4.4.6).
- **Applications.** In the United States, the placement of medical school graduates into residency positions (NRMP) is carried out with a version of this algorithm. The Boston and New York school choice systems are also based on deferred acceptance. Roth and Shapley received the 2012 Nobel Prize in Economics for this work.

When all six matchings are tried with the preferences in the example, only one stable matching is found: A–Y, B–X, C–Z. So the result would be the same if the projects proposed. When the projects propose, X's first choice is B, and Y's and Z's first choice is A. A keeps Y between Y and Z; Z is rejected and proposes to its next choice B. B rejects Z because B prefers X to Z; Z finally proposes to C. The result is again A–Y, B–X, C–Z.

Common mistake

Implementing deferred acceptance as "first come, first served". Y's acceptance of C in round 1 is not final; when A arrives in round 2, C is dropped. An implementation that treats acceptance as final (for example, Y keeps C and rejects A) can lead to an unstable matching: A and Y want each other but stay apart.

4.4.6 The receiving side can gain by lying

Gale–Shapley is DSIC for the proposers but not for the receiving side. A new example shows this. The students propose again. The preferences are: A: $X \succ Y \succ Z$, B: $Y \succ X \succ Z$, C: $X \succ Y \succ Z$. The projects' true preferences are X: $B \succ A \succ C$; Y and Z: $A \succ B \succ C$.

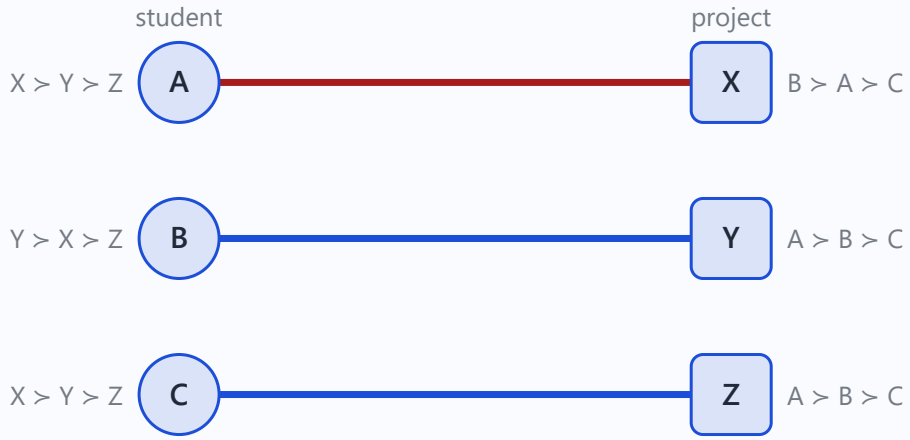


Figure 4.17: Under honest reporting the result is A–X, B–Y, C–Z. X is matched with A, its true 2nd choice (red line).

Example 4.16

Honest reporting.

1. A→X: X is empty, A is held. B→Y: Y is empty, B is held.
2. C→X: X prefers A to C; C is rejected. C→Y: Y prefers B to C; C is rejected. C→Z: accepted.
3. Result: A–X, B–Y, C–Z. X gets A, who is 2nd in its true order.

X lies and reports the order B ≻ C ≻ A; it puts A last.

1. A→X: X is empty, A is held. B→Y: B is held.
2. C→X: in the order X reported, C is better than A. X keeps C and rejects A.
3. A→Y: Y prefers A to B. A is held, B is rejected.
4. B→X: X prefers B to C. B is held, C is rejected.
5. C→Y: Y prefers A to C; C is rejected. C→Z: accepted.
6. Result: A–Y, B–X, C–Z. X now has its true 1st choice B. The lie paid off.

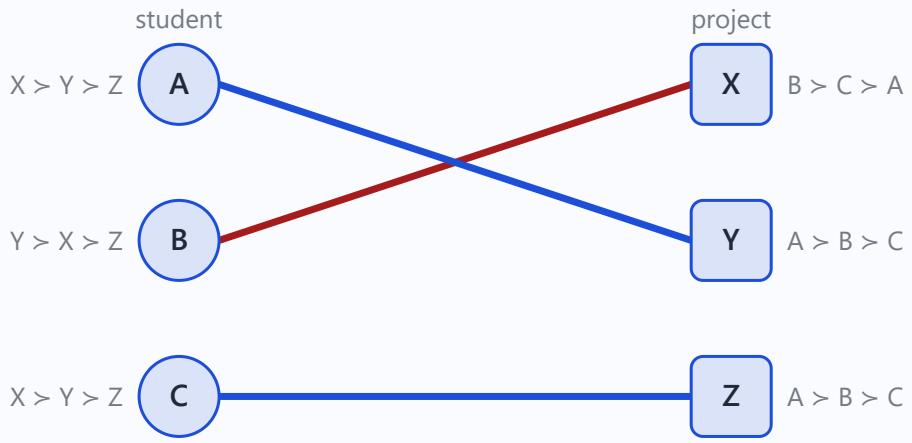


Figure 4.18: After X's lie the result is A–Y, B–X, C–Z. A's rejection by X sets off a chain that brings B to X; X is matched with its true 1st choice B (red line).

How the lie works can be seen by comparing Figure 4.17 with Figure 4.18. By rejecting A, X starts a *rejection chain*. The rejected A goes to Y and takes B's place there. B, displaced, comes around to its next choice X. X's lie has built a chain that brings the student it wants back to itself.

Checking the six matchings shows that these preferences have two stable matchings: A–X, B–Y, C–Z, which is best for the students, and A–Y, B–X, C–Z, which is best for the projects. Student-proposing Gale–Shapley finds the first. X's lie moves the result to the second, the one best for the projects. This is a special case of a general result: no stable mechanism can be DSIC for both sides at once (Roth,

1982). The designer has to choose which side to guarantee honesty for. In school placement the students are usually made the proposing side; the schools' preferences mostly come from legal priority rules and cannot be changed strategically.

4.4.7 Facility location: mean or median?

Five villages, A, B, C, D, and E, lie on a straight road at kilometers 2, 5, 9, 14, and 20, respectively. A single facility, for example a hospital, is to be built along the road. Each village wants the facility to be close to it; a village's cost is its distance to the facility. The villages' locations are treated as private information: each village reports its own location (or its most preferred point) and the facility is placed according to the reports.

The preferences in this setting have a special structure. Each village has a single favorite point on the road (its own location), and its satisfaction falls as the facility moves away from that point. Preferences of this kind are called **single-peaked preferences**: the satisfaction graph has a single peak. The Gibbard–Satterthwaite theorem does not assume this structure; a strategy-proof rule that is not a dictatorship can therefore be found here.

Two rules are compared. The **mean mechanism** places the facility at the average of the reports. The **median mechanism** sorts the reports and picks the middle one. With five reports, the median is the third number in the sorted order.

Case	Reports	Facility	E's distance
Mean, everyone honest	2, 5, 9, 14, 20	10	10
Mean, E reports "40"	2, 5, 9, 14, 40	14	6
Median, everyone honest	2, 5, 9, 14, 20	9	11
Median, E reports "40"	2, 5, 9, 14, 40	9	11
Median, E reports "8"	2, 5, 8, 9, 14	8	12

Table 4.9: Village E's (km 20) reports under the mean and median rules.

Example 4.17

1. **Mean, everyone honest.** $(2 + 5 + 9 + 14 + 20)/5 = 50/5 = 10$. The facility is built at km 10; E's distance is $20 - 10 = 10$ km.
2. **Mean, E lies.** E reports its location as 40: $(2 + 5 + 9 + 14 + 40)/5 = 70/5 = 14$. The facility moves to km 14; E's distance falls to $20 - 14 = 6$ km. The lie paid off: the mean can be manipulated.
3. **E's most profitable lie.** If E wants to pull the facility exactly to its own location, the mean must be 20: $(2 + 5 + 9 + 14 + x)/5 = 20$, that is, $30 + x = 100$ and $x = 70$. By reporting 70, E reduces its distance to 0.

4. **Median, everyone honest.** The sorted reports are 2, 5, 9, 14, 20; the middle one is 9. E's distance is *20 − 9 = 11* km.
5. **Median, E reports 40.** The order becomes 2, 5, 9, 14, 40; the median is still 9. An extreme value cannot pull the median; E's distance stays at 11.
6. **Median, E reports 8.** If E crosses to the other side of the median, the order becomes 2, 5, 8, 9, 14 and the median falls to 8. E's distance rises to *20 − 8 = 12* km. The lie did harm.

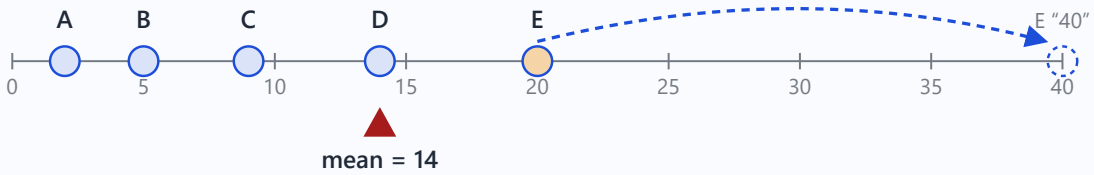


Figure 4.19: Under the mean rule, E reports its location as 40 instead of 20 (dashed arrow); the mean moves from 10 to 14 and the facility gets closer to E.

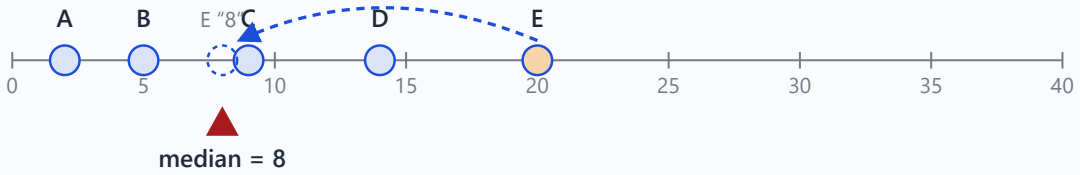


Figure 4.20: Under the median rule, E crosses to the other side of the median and reports 8; the median falls from 9 to 8 and the facility moves away from E.

The weakness of the mean is that every report affects the result directly. If a village raises its report by 5 units, the mean rises by 1 unit. A village at the edge can pull the mean wherever it wants by exaggerating its location. The median, on the other hand, looks only at the ranking. A village to the right of the median stays on the right however much it inflates its report, and the middle element of the ranking does not change.

The median has another advantage. The total distance is *7 + 4 + 0 + 5 + 11 = 27* at the median and *8 + 5 + 1 + 4 + 10 = 28* at the mean. The median is the point that minimizes the total distance. The reason is this: if the facility is at a point to the right of the median, moving it one unit to the left brings each village on the left one unit closer and takes it one unit farther from each village on the right. To the right of the median, more villages are on the left than on the right, so the move reduces the total. The same argument works from the left. The total can be reduced no further only at the median. So the median is both strategy-proof and best in terms of total cost. When all reports between 0 and 60 are tried for each village on a computer, no lie is found to pay off under the median.

4.4.8 Why is the median strategyproof?

Theorem 4.8 (Median mechanism)

With single-peaked preferences (here: each village wants to minimize its distance to the facility), the median mechanism (the middle report when the number of villages is odd; a predetermined one of the two middle reports, for example the left one, when it is even) is strategy-proof: for each village, reporting its true location is a dominant strategy (Moulin, 1980).

The proof consists of three cases, depending on where the village stands relative to the median. Let village i's true location be x_i, the median when the others are honest m, and the village's false report r. The argument holds whatever the others report; to make it concrete, the villages of the example are used.

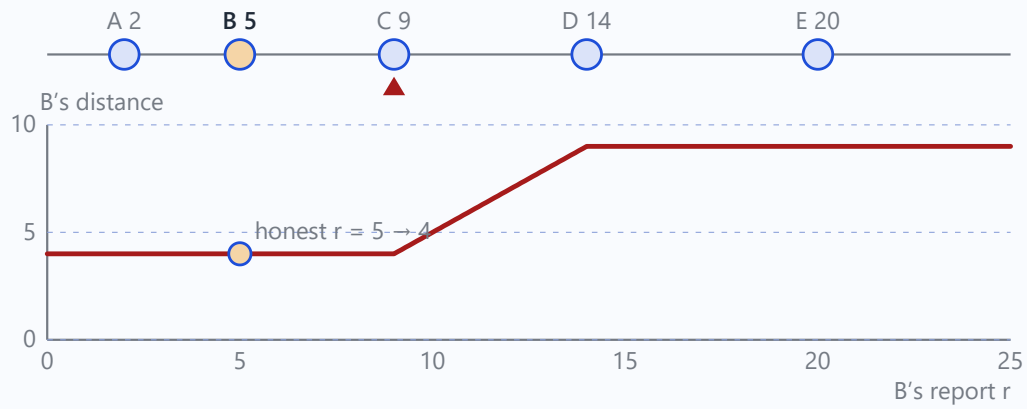


Figure 4.21: Case 1: B (km 5) is to the left of the median. The graph shows B's distance to the facility as a function of B's report r. For $r \le 9$ the distance is constant at 4; $r > 9$ pushes the median to the right and the distance grows up to 9.

Proof. **Case 1: $x_i < m$.** In the example B = 5 and the median is 9. When B is honest the distance is $9 - 5 = 4$. Any report by B that is less than or equal to 9 keeps B to the left of (or at) the median in the ranking; the median stays at 9 and the distance is 4. A report by B greater than 9 moves B to the right of the median; the median then shifts right (to B's report or at most to 14) and away from B. No report by B can pull the median to the left, toward B, because there are already B and A to the left of the median, and B changing its place there does not change the middle of the ranking. No r can bring the distance below 4.

Case 2: $x_i > m$. In the example D = 14. This case is the mirror image of the first. Any report by D greater than or equal to 9 leaves the median unchanged; the distance stays at $14 - 9 = 5$. A report by D less than 9 pulls the median left, away from D.

Case 3: $x_i = m$. In the example C = 9. The facility is exactly at C's location and the distance is 0. This is the smallest possible distance; no lie can do better. Any lie can only move the median away from C or leave it in place.

In all three cases the honest report gives the best result, whatever the others report. The median mechanism is strategy-proof. □

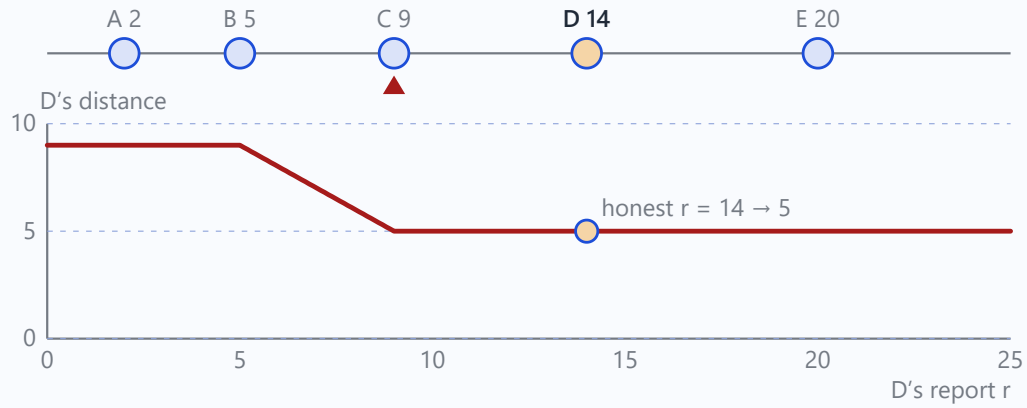


Figure 4.22: Case 2: D (km 14) is to the right of the median. For $r \geq 9$ the distance is constant at 5; $r < 9$ pulls the median to the left and the distance grows up to 9.

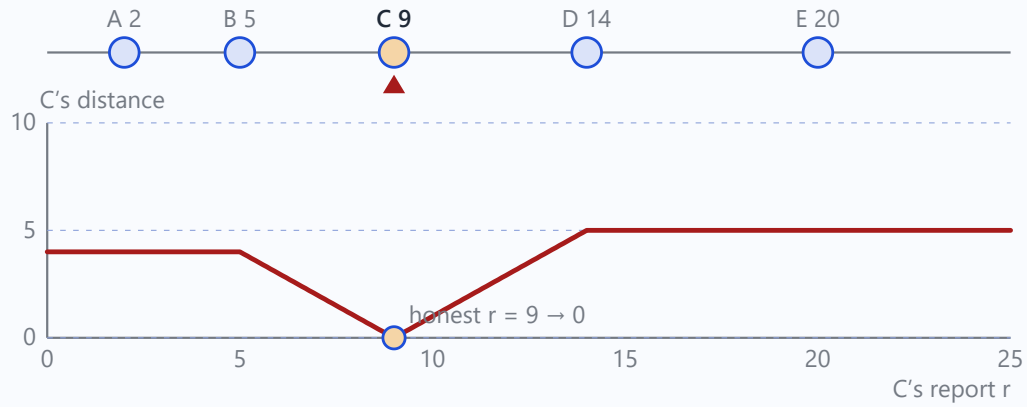


Figure 4.23: Case 3: C (km 9) is at the median. Under honest reporting the distance is 0; any report that moves away from 9 also moves the median away from C (at most to km 5 or km 14).

In all three graphs the distance curve never falls below the honest report: the flat parts are reports where the lie changes nothing, and the rising parts are reports where the lie does harm.

The same idea carries over to voting. If the alternatives can be ordered along a line (for example candidates on a left–right axis, or budget sizes) and each voter has a single-peaked preference, the median voter's favorite alternative beats every other alternative in a pairwise vote. This result is known as Duncan Black's median voter theorem. One way to escape the bad news of Gibbard–Satterthwaite is to recognize the cases in which preferences have this structure.

Common mistake

Taking the median to be "the average of the two middle ones" when the number of villages is even. This brings back the manipulability of the mean: one of the two middle villages can change its report and pull the facility toward itself. For strategy-proofness, with an even number of reports the rule must pick one of the two middle ones (for example the left one).

4.4.9 Serial dictatorship and the Boston mechanism

In school placement, students rank schools and each school has limited capacity. Schools may also have **priority** orders over students: for example, those who live near the school or have a sibling there may get priority. Two simple mechanisms create very different incentives in this setting.

Serial dictatorship. Students are lined up in an order. The student whose turn it is takes their favorite among the remaining schools. A student cannot affect the choices of those before them, and their own choice depends only on their own preference list. Reporting true preferences is therefore a dominant strategy: the mechanism is DSIC. The outcome is also **Pareto optimal**: no other assignment can make someone better off without making anyone worse off. If the order is determined by lottery, the mechanism is called **random serial dictatorship** (RSD). It differs from the dictatorship of Gibbard–Satterthwaite in that the "dictator" is not a single person; the order gives everyone a turn to choose.

The Boston mechanism, or **immediate acceptance**. In each round, unplaced students apply to the next school on their list. Each school gives its open seats to the students applying to it in that round, in priority order. A seat once given is not taken back in later rounds. This is where it differs from deferred acceptance: acceptance is final, not tentative. A student who loses their first choice may therefore find their second choice already full when applying to it.

In the example there are three students (A, B, C) and three schools (X, Y, Z) with one seat each. The preferences are A: $X \succ Y \succ Z$, B: $X \succ Y \succ Z$, C: $Y \succ X \succ Z$. The priorities are A, B, C for X and Z, and B, A, C for Y.

Round	B honest ($X \succ Y \succ Z$)	B reports "$Y \succ X \succ Z$"
1	A, B apply to X → A gets it; C applies to Y → C gets it	A applies to X → A gets it; B, C apply to Y → B gets it
2	B → Y: full	C → X: full
3	B → Z: B gets it	C → Z: C gets it
Outcome	A–X, B–Z, C–Y	A–X, B–Y, C–Z

Table 4.10: B's honest and strategic reports under the Boston mechanism.

Example 4.18

1. **B honest, round 1.** A and B apply to their first choice X. In X's priority order A is ahead of B; X is given to A. C applies to its first choice Y and, as the only applicant, gets Y. These seats are now final.
2. **Round 2.** The only unplaced student is B. B applies to its second choice Y, but Y was given to C in round 1.
3. **Round 3.** B applies to its third choice Z and gets Z. The result is A–X, B–Z, C–Y. B ended up with its last choice.
4. **B lies.** B writes its list as "Y $\succ$ X $\succ$ Z". In round 1 only A applies to X and gets X. B and C apply to Y; in Y's priority order B is ahead of C, so Y is given to B.
5. **Rounds 2 and 3.** C applies to its second choice X; X is full. C is placed at Z. The result is A–X, B–Y, C–Z.
6. **Comparison.** B's true order is X $\succ$ Y $\succ$ Z. When honest B got Z; by lying it got Y. Putting its second choice at the top of the list paid off: the Boston mechanism is not DSIC.

The reason for the lie is final acceptance. If B guesses that it cannot get X anyway, it does not want to "waste" the first round, because in the first round Y's seat will be filled. Under the Boston mechanism, students have to guess their chances and arrange their preference lists accordingly. This gives an advantage to families with access to information and advice, while families who write their preferences as they are may lose out.

On the same profile, student-proposing Gale–Shapley with honest reports gives A–X, B–Y, C–Z. In round 1, A and B propose to X and C to Y; X keeps A and rejects B. B proposes to Y; since B is ahead of C in Y's priority order, B is held and C is rejected. C loses to A at X and is placed at Z. Under deferred acceptance, C's seat at Y in round 1 was tentative, so B can get Y by staying honest. Abdulkadiroğlu and Sönmez (2003) showed that writing preferences as they are can hurt a student under the Boston mechanism, and brought the mechanism design viewpoint to school choice. The city of Boston switched to student-proposing deferred acceptance for school placement in 2005.

4.4.10 Swapping dorm rooms: TTC

This time each student holds a dorm room and no money is used. Students rank other students' rooms along with their own. The goal is to find a redistribution, through voluntary swaps, in which everyone is at least as well off as with what they hold. This problem is called the **housing market** problem. The problem was defined by Shapley and Scarf (1974), who attributed the following algorithm to David Gale.

The **Top Trading Cycles** (TTC) algorithm works as follows:

1. Each remaining person points to the owner of their favorite room among the remaining rooms (draws an arrow). If they like their own room best, they point to themselves.

2. Since exactly one arrow leaves each person, this graph contains at least one cycle. Everyone in a cycle gets the room they point to and leaves the market with their room.
3. Repeat until no one is left.

The claim about a cycle in the second step rests on a simple observation. Start from any person and follow the arrows; since there are finitely many people, at some point we return to someone already visited. This return forms a cycle. A cycle can also consist of a single person: a person who points to their own room keeps it.

Person	Preference (room owner)
A	C ≻ A ≻ B
B	A ≻ D ≻ B
C	B ≻ A ≻ C
D	C ≻ E ≻ B ≻ D
E	A ≻ D ≻ E

Table 4.11: Room preferences of five students. Rooms are referred to by their owner's name; rooms not on the list are worse than the person's own room.

Example 4.19

1. **Round 1, arrows.** Everyone points to the owner of their favorite room among those remaining: A→C, B→A, C→B, D→C, E→A.
2. **Round 1, cycle.** Start at A and follow the arrows: A→C, C→B, B→A. We are back at A; the cycle is A → C → B → A.
3. **Round 1, trade.** Everyone in the cycle gets the room they point to: A gets C's room, C gets B's room, B gets A's room. All three got their first choice and leave with their rooms. The arrows of D and E (to C and to A) were not in the cycle; they stay.
4. **Round 2, arrows.** D and E remain. D's first choice, C, is gone; the next is E: D→E. E's first choice, A, is gone; the next is D: E→D.
5. **Round 2, cycle and trade.** D → E → D is a cycle. D gets E's room and E gets D's room. No one is left.
6. **Result.** A ← C's room, B ← A's room, C ← B's room, D ← E's room, E ← D's room. A, B, and C traded in a three-way cycle; D and E in a two-way cycle.

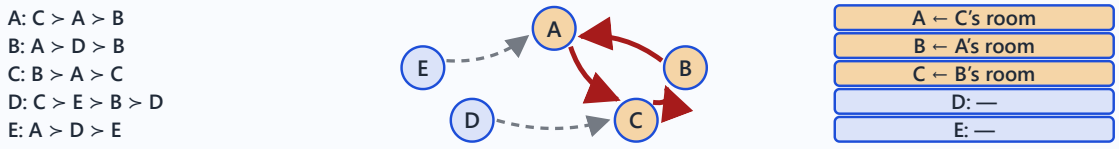


Figure 4.24: TTC, round 1. The bold red arrows form the cycle A → C → B → A; those in the cycle get the rooms they point to (list on the right). D's and E's dashed arrows are not in the cycle, so both are left for the next round.

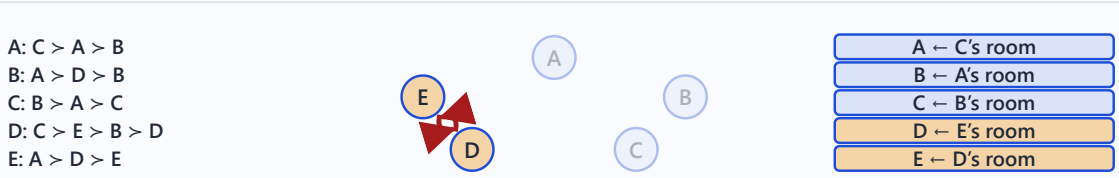


Figure 4.25: TTC, round 2. The remaining D and E point to each other; they trade in the cycle D → E → D. The list on the right shows the final assignment.

TTC has three important properties:

- **DSIC.** Reporting true preferences is a dominant strategy for every person (Roth, 1982). A person gets their favorite among the remaining rooms in the round in which they enter a cycle. The rooms of those who left earlier cannot be changed by this person's report, because those cycles formed independently of the person's arrow.
- **Pareto optimality.** No other assignment can make someone better off without making anyone worse off. Those who left in the first round got their favorite room; improving on that is impossible. Those who left in a later round got their favorite among those remaining; to improve on that they would need the room of someone who left earlier, and this would make that person worse off.
- **The core.** The outcome is **in the core**: no group, by trading only their own rooms among themselves, can reach an outcome that is at least as good for every member and better for at least one (Roth and Postlewaite, 1977). In particular, everyone gets a room at least as good as their own; IR holds.

When all possible false preference lists of each person in the example are tried on a computer, none gives a better room. The same rule also fits swaps of duty or shifts. Its most impressive real-world application is kidney exchange: patients with an incompatible donor trade donors in cycles (Roth, Sönmez, and Ünver, 2004).

Common mistake

Leaving the rooms of those who exited in one round as targets of arrows in later rounds. In round 2, D's first choice is C's room, but C's room went to A in round 1 and left the market. D now chooses from those remaining, that is, between E's room and its own. The arrows are

redrawn in every round among those remaining only.

4.4.11 Code trace: ttc()

The Python version of TTC collects the arrows in a dictionary in each round, finds a cycle by starting from one person and following the arrows, and removes those in the cycle. The input is the dictionary `pref`: `pref[i]` is the list that ranks the room owners from best to worst for `i`, and it includes their own name.

```
def ttc(pref):
  left, assign = list(pref), {}
  while left:
    g = {i: next(j for j in pref[i]
             if j in left) for i in left}
    i, path = left[0], []
    while i not in path:          # follow arrows
      path.append(i); i = g[i]
    cycle = path[path.index(i):]
    for k in cycle:               # trade
      assign[k] = g[k]; left.remove(k)
  return assign
```

The parts of the code are:

- `left` is the list of people who have not yet traded, and `assign` is the result dictionary: `assign[k]` is the owner of the room that `k` receives.
- The dictionary `g` holds this round's arrows: for each remaining person `i`, the first person in the list `pref[i]` who is among those remaining. `next(...)` returns the first element that satisfies the condition.
- `i = left[0]` starts from the first remaining person. The inner loop adds `i` to the path if it has not been added before and follows its arrow (`i = g[i]`). It stops when it reaches someone seen before.
- `cycle = path[path.index(i):]` is the cyclic part of the path: from the place where the repeated person first appeared to the end. The people at the start of the path may be people who lead into the cycle but are not on it; this slicing leaves them out.
- Each person in the cycle gets the room they point to and is removed from `left`.

Round	left	g (arrows)	path	cycle	assign (added)
1	A, B, C, D, E	A→C, B→A, C→B, D→C, E→A	[A, C, B], then i = A	[A, C, B]	A: C, C: B, B: A
2	D, E	D→E, E→D	[D, E], then i = D	[D, E]	D: E, E: D
end	empty	return: {A: C, C: B, B: A, D: E, E: D}			

Table 4.12: Trace of `ttc(pref)`.

In this example the path began directly with the cycle in every round. With other inputs the path may start at someone who is not in the cycle. For example, if the trace in round 1 had started from D, the path would be D → C → B → A → C; it stops when C is seen the second time, and `path.index('C')` separates the cycle [C, B, A]. D is not in the cycle and stays. Since at least one person leaves in each round, there are at most n rounds; building the arrows takes at most n^2 work in each round. In this simple version the algorithm finishes in $O(n^3)$ time.

4.5 Bayesian games

So far the private information has been between the designer and the players: the designer did not know the types. But the players often do not know one another's types either. In an auction a bidder does not know the rivals' values; in a negotiation the buyer does not know the seller's cost. To study such situations of incomplete information, game theory uses Bayesian games. Bayesian games are also the framework on which the BIC concept (4.2.4) rests.

4.5.1 Incomplete information and Bayesian games

Solving a game of incomplete information looks impossible at first sight: a player does not even know which game the rival is playing. John Harsanyi (1967–68) solved this problem with an elegant idea. A "chance" or "Nature" move, outside the players, is added to the game. At the start of the game, Nature chooses each player's type according to a probability distribution. Each player sees only their own type. The distribution itself is known to everyone. The problem of an "unknown game" thus becomes the problem of "a known game with chance in it". This method is called the **Harsanyi transformation**.

> **Definition 4.9**
>
> A **Bayesian game** consists of the following elements: the players; for each player an action set and a type set Θ_i; a **common prior** π over type profiles, known to everyone; and, for each player, a payoff function $u_i(a, \theta)$ that depends on the action profile a and the type profile θ.

In a Bayesian game the notion of strategy changes. A player decides after learning their own type; but the person doing the analysis (and the rivals) does not know the player's type. A strategy is therefore a **rule** that says which action to choose for each type. For example, the strategy of a player with two types is a mapping such as "if I am of type S, play C; if I am of type A, play M". The rival knows this rule but does not know which line will be applied.

Definition 4.10

A strategy profile is a **Bayes–Nash equilibrium** if every player, at each of their types, maximizes their expected payoff against the other players' strategies. The expected payoff is taken over the types the player does not know, using the common prior.

The definition is a natural extension of Nash equilibrium. In Nash equilibrium we say "no one wants to deviate alone". In Bayes–Nash equilibrium the same is said, except that the gain from deviating is measured by expected value and each type is checked separately. If a player has two types, each of the two types must play its own best response.

As an example, consider a meeting game. P1 will go in the evening either to a concert (C) or to a match (M) and prefers the concert. P2 is one of two types: **type S** wants to meet P1, and **type A** wants to avoid P1. P1 does not know P2's type; each type has probability 1/2. P2 knows their own type.

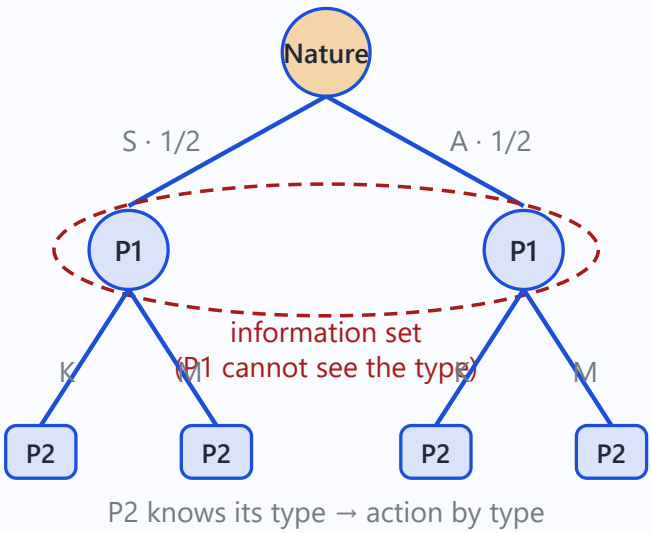


Figure 4.26: The meeting game after the Harsanyi transformation. Nature first chooses P2's type: S (1/2) or A (1/2). P1 does not know which branch it is on; the dashed ellipse joining the two nodes is P1's information set. P2 knows its own type and chooses an action by type.

Figure 4.26 shows the game tree. The dashed ellipse shows that P1 cannot tell the two nodes apart: P1 must choose the same action without knowing which type of P2 it is playing against. This is called P1's **information set**. P2, on the other hand, decides separately on each branch, so it can have a type-dependent strategy.

The Bayesian version of the revelation principle links this to mechanism design. Every outcome that can be implemented in a Bayes–Nash equilibrium can also be implemented by a direct mechanism in which honest reporting is a Bayes–Nash equilibrium, that is, by a BIC direct mechanism. The proof is the same as in 4.2.3; the only difference is that the sentence "the deviation is not profitable" is stated for expected payoffs.

4.5.2 Finding a Bayes–Nash equilibrium

The payoffs of the game are given by two matrices: one for when P2 is of type S, the other for type A. Each cell shows (P1's payoff, P2's payoff). P1's payoffs are the same in both matrices: in every case P1 wants to meet P2, at the concert or at the match, and prefers the concert (meeting at C is 2, meeting at M is 1, failing to meet is 0). What changes is P2's payoffs.

P1 \ P2	C	M
C	(2, 1)	(0, 0)
M	(0, 0)	(1, 2)

Table 4.13: P2 is of type S (wants to meet), probability 1/2.

P1 \ P2	C	M
C	(2, 0)	(0, 2)
M	(0, 1)	(1, 0)

Table 4.14: P2 is of type A (wants to avoid), probability 1/2.

Counting the pure strategies shows how large the search is. P1 has two pure strategies: C or M. Since each type of P2 can choose C or M, P2 has $2 \times 2 = 4$ pure strategies: (S→C, A→C), (S→C, A→M), (S→M, A→C), (S→M, A→M). There are $2 \times 4 = 8$ pure strategy profiles in all. Instead of checking them all one by one, we take a shorter route: for each choice of P1, find the best response of each type of P2, then see whether P1 deviates against those responses.

Example 4.20

1. **Candidate 1: P1 = C.** P2's best response depends on the type. Type S, when P1 is at C, gets 1 at C and 0 at M; it chooses C. Type A, when P1 is at C, gets 0 at C and 2 at M; it chooses M. P2's strategy is s_2 = (S→C, A→M).
2. **Does P1 deviate?** While P2 plays this strategy, P1's expected payoff is: if it chooses C, with probability one half it meets type S at C (2) and with probability one half type A runs off to M (0): $\frac{1}{2} \cdot 2 + \frac{1}{2} \cdot 0 = 1$. If it chooses M, it fails to meet type S, who is at C (0), and meets type A, who is at M (1): $\frac{1}{2} \cdot 0 + \frac{1}{2} \cdot 1 = 0.5$. $1 > 0.5$; C is best for P1.
3. **Result.** (C; S→C, A→M) is a Bayes–Nash equilibrium.
4. **Candidate 2: P1 = M.** Type S, when P1 is at M, gets 0 at C and 2 at M; it chooses M. Type A, when P1 is at M, gets 1 at C and 0 at M; it chooses C. s_2 = (S→M, A→C).
5. **Does P1 deviate?** If it chooses C, type S is at M (0) and type A is at C (2): $\frac{1}{2} \cdot 0 + \frac{1}{2} \cdot 2 = 1$. If it chooses M, type S is at M (1) and type A is at C (0): $\frac{1}{2} \cdot 1 + \frac{1}{2} \cdot 0 = 0.5$. C is better for P1; it deviates from M to C. This is not an equilibrium.

6. **The remaining profiles.** With one of P1's choices fixed, each type of P2 has a unique best response (no ties). Another strategy of P2 would therefore fail to be a best response at some type, and those profiles cannot be equilibria. Checking all 8 profiles shows that the only pure Bayes–Nash equilibrium is (C; S→C, A→M).

Candidate profile	P1: C expected	P1: M expected	Equilibrium?
P1 = C; S→C, A→M	½·2 + ½·0 = 1	½·0 + ½·1 = 0.5	✓ yes
P1 = M; S→M, A→C	½·0 + ½·2 = 1	½·1 + ½·0 = 0.5	✗ deviates to C

Table 4.15: Candidate profiles and P1's expected payoffs (P(S) = 1/2).

The interpretation is interesting. If P1 goes to M, type S follows and type A runs off to the concert. But the concert, where type A runs to, is the place P1 likes more. Since the chance of meeting is the same (one half in both candidates) and P1 prefers the concert, P1 wants to stay at C. In equilibrium P2's behavior gives away its type: a P2 seen at the concert is type S, and one seen at the match is type A.

Common mistake

Mistaking P2's strategy for a single action. "P2 plays C" is not a Bayesian strategy; what each type will do must be stated separately. Also, when computing P1's expected payoff, each type's *own* choice must be used: in the candidate P1 = C, type S is at C and type A is at M; putting both types in the same place gives a wrong result.

4.5.3 How the equilibrium changes with the probability

If the prior changes, the equilibrium can change too. Let the probability that P2 is of type S be a general number π; the probability of type A is $1 - \pi$. P2's best responses do not depend on π, because P2 knows its own type. Only P1's expected payoffs change.

- **Candidate 1 (P1 = C; S→C, A→M).** If P1 stays at C it gets $\pi \cdot 2 + (1 - \pi) \cdot 0 = 2\pi$; if it switches to M it gets $\pi \cdot 0 + (1 - \pi) \cdot 1 = 1 - \pi$. For equilibrium we need $2\pi \geq 1 - \pi$, that is, $\pi \geq 1/3$.
- **Candidate 2 (P1 = M; S→M, A→C).** If P1 stays at M it gets $\pi \cdot 1 = \pi$; if it switches to C it gets $(1 - \pi) \cdot 2$. For equilibrium we need $\pi \geq 2(1 - \pi)$, that is, $\pi \geq 2/3$.

Range of π	Candidate 1	Candidate 2	Number of pure equilibria
π < 1/3	✗	✗	0 (a mixed equilibrium is needed)
1/3 ≤ π < 2/3	✓	✗	1
π ≥ 2/3	✓	✓	2

Table 4.16: Pure Bayes–Nash equilibria as a function of the probability π of type S.

$\pi = 1/2$ falls in the second row; this is why a single pure equilibrium was found in the example. If type S is very unlikely ($\pi < 1/3$), P2 is very probably of type A and runs from P1 wherever P1 goes, so P1 wants to switch to the other place every time. In this case there is no pure equilibrium; by Nash's existence theorem there is an equilibrium in mixed strategies. If type S is very likely ($\pi \geq 2/3$), the game comes close to a coordination game and two pure equilibria appear.

At the threshold values $\pi = 1/3$ and $\pi = 2/3$, P1 is indifferent between its two actions. For example, at $\pi = 1/3$ in candidate 1, both C and M give $2/3$. In the case of a tie, P1 staying at C is a best response; this is why the inequalities are written with "≥".

4.5.4 Mechanisms from a Bayesian viewpoint

Bayesian games are used in mechanism design where DSIC is not enough or is too strict. Four examples show why this viewpoint is needed.

The first-price auction. In a first-price auction, an honest bid leaves the winner with zero utility (4.2.4). In equilibrium, therefore, everyone bids below their value. How far below? The answer depends on beliefs about the rivals' values; the equilibrium is therefore Bayesian. In a standard example the values of two bidders are uniformly distributed between 0 and 1 (every value is equally likely). If the rival bids half its value, the probability that a bidder with value v wins with a bid b is the probability that the rival's value is below $2b$, that is, $2b$ (for $b \leq 1/2$). The expected utility is $(v - b) \cdot 2b$. This expression is largest at $b = v/2$. So the strategy "bid half your value" is a best response to itself for two players: it is a Bayes–Nash equilibrium.

Example 4.21

The revelation principle builds a direct mechanism from the first-price equilibrium: "Report your value; the highest report wins and pays half of its report." If values are uniform between 0 and 1, this mechanism is BIC. An example shows that it is not DSIC.

1. Let the bidder's value be 0.8 and let the rival report 0.9. An honest report (0.8) loses; the utility is 0.
2. If the bidder reports 0.95, they win and pay $0.95/2 = 0.475$. The utility is $0.8 - 0.475 = 0.325 > 0$.
3. Against this rival report, lying paid off. Honesty is not best whatever the rival reports; the mechanism is not DSIC. But when the rival is honest and values are uniform, honesty is best on average.

This example shows why the converse of the relation DSIC ⇒ BIC is not true in general.

DSIC or BIC? DSIC is independent of the players' beliefs and is robust: even if the type distribution is estimated wrongly, honesty remains best. BIC is weaker but allows more goals. For example, VCG cannot achieve strong budget balance in the public goods problem; yet at the BIC level there is a

public goods mechanism that is both efficient and strongly budget balanced (d'Aspremont and Gérard-Varet, 1979; Arrow, 1979). In this mechanism the payments are adjusted according to the *expected* effect of each player's report on the others, and the money collected is redistributed to the players.

Maximum revenue. The auction that maximizes the seller's expected revenue is also a Bayesian question, because "expected revenue" depends on the distribution of the buyers' values. Myerson (1981) solved this question. Under standard conditions the answer is an auction with a **reserve price**: the seller sets a minimum price, and no sale is made below it. A reserve price sometimes causes no sale at all, which hurts efficiency, but in return it raises the expected revenue. This trade-off between efficiency and revenue is one of the basic topics of auction design.

Poker and bluffing. In poker the rival's hand (type) is hidden. Bluffing, that is, a large bet with a weak hand, is part of a mixed Bayesian strategy: if strong hands are always played with a large bet, the size of the bet gives the hand away.

4.6 Review

The assumptions and limits of mechanism design, a comparison of the chapter's mechanisms, and a few short problems are collected below.

4.6.1 Behavior, fairness, and ethics

Do people always respond to incentives? The theory assumes that players maximize their utilities. In reality, emotions, morality, habit, and a sense of fairness also influence decisions. The rejection of low offers in the Ultimatum Game (4.2.5) is the best-known example. Incentives are strong but not the only determinant. This observation does not invalidate mechanism design; it brings two cautions. First, the utility function should reflect what people actually care about. Second, rules should be simple and understandable. For example, the English auction gives the same outcome as Vickrey, but it is much easier to see there why honest behavior is best.

Ethical problems in mechanism design. Even good rules can lead to bad outcomes:

- **Manipulation and withheld information.** Under rules that are not DSIC, those with access to information and advice have an advantage. Under the Boston mechanism, families who know how to write strategic lists can get ahead of families who write their preferences as they are.
- **The advantage of powerful players.** Collusion and false identities give organized groups power that the rule did not foresee.
- **Unfair distribution of income.** Efficiency maximizes the sum of values; but value depends on ability to pay. A rich player's "high value" often shows only that they have more money.
- **Excluding those who cannot pay.** Mechanisms that use money shut out those who cannot pay from the start. This is why money is not used in areas such as school placement and organ transplantation.

Ethical limits must therefore be set by society and its institutions. A good mechanism must be not only efficient but also fair and ethical. Mechanism design is a tool in this debate: it shows clearly which goal can be achieved with which rules and at what cost; which goal to pursue is for society to decide.

4.6.2 The chapter's mechanisms side by side

The mechanisms seen in the chapter are compared in Table 4.17. The table shows two distinctions together: whether money is used, and at what level the incentive property holds.

Mechanism	Money	Incentive property	What it provides
Proportional cost sharing	w_in	Not IC (free riding)	—
VCG (Clarke pivot)	w_in	DSIC, IR	Efficient outcome; may run a budget surplus
Vickrey (VCG for one item)	w_in	DSIC, IR	Highest value wins
Gale–Shapley	w_out	DSIC for proposers	Stable matching, $O(n^2)$
Median (facility location)	w_out	DSIC (strategy-proof)	Minimum total distance
Serial dictatorship	w_out	DSIC	Pareto optimal assignment
Boston (immediate acceptance)	w_out	Not DSIC	Priority to first choices; strategic lists are rewarded
TTC (housing market)	w_out	DSIC	Pareto optimal, in the core
Bayesian mechanisms	w_in	BIC (Bayes–Nash)	Expected-value goals (e.g. revenue)

Table 4.17: The mechanisms of this chapter.

The main lesson of the table is this: thanks to the revelation principle, it is enough to focus on direct mechanisms with honest reporting; but IC, IR, efficiency, and budget balance cannot always be achieved together. With money, VCG achieves efficiency and DSIC and gives up budget balance. Without money, Gibbard–Satterthwaite rules out a general solution; the solution comes from the structure of preferences (median) or from changing the goal (stability, trade).

4.6.3 Practice problems

The following five problems bring together the chapter's computational methods in short situations.

Example 4.22

Two identical items, three bidders. Each bidder wants at most one item; the values are 10, 8, and 5. Under VCG, who wins and who pays what?

1. **The efficient outcome.** The two items go to the two highest values: 10 and 8 win; the welfare is 18.

2. **Payment of the bidder with value 10.** Without this bidder the items would go to 8 and 5: the others' best welfare is $8 + 5 = 13$. With this bidder present, the others get only the item of the bidder with value 8: welfare 8. The payment is $13 - 8 = 5$.
3. **Payment of the bidder with value 8.** Without this bidder the others' welfare would be $10 + 5 = 15$; with this bidder, 10. The payment is $15 - 10 = 5$.
4. **The bidder with value 5.** The outcome is the same without this bidder; the payment is 0.

Each winner pays the highest value among the losers (5). This is the generalization of the single-item second-price rule to several identical items.

Example 4.23

Bora's most profitable report. In the proportional cost sharing example (park 60; values 30, 25, 20), what should Bora report if Ayşe and Cem are honest?

1. The sum of Ayşe's and Cem's reports is $30 + 20 = 50$. For the park to be built, Bora must report at least 10.
2. If Bora reports 10, the total is exactly 60 and Bora pays $60 \cdot 10/60 = 10$. The utility is $25 - 10 = 15$.
3. When honest the utility was 5. The lie gains 10 units; the rule is not IC.

Example 4.24

Villages at 3, 7, 8, 12, 30. What are the median and the mean? Under the mean rule, what should the village at km 30 report to pull the facility exactly to itself?

1. In the sorted list the middle number is 8: the median is 8.
2. The mean is $(3 + 7 + 8 + 12 + 30)/5 = 60/5 = 12$.
3. For the mean to be 30 we need $(3 + 7 + 8 + 12 + x)/5 = 30$: $30 + x = 150$, $x = 120$. By reporting 120 the village pulls the facility to its own location. Under the median, a report of 120 leaves the median at 8.

Example 4.25

If the probability of type S were 1/4, would (C; S→C, A→M) still be a Bayes–Nash equilibrium?

1. P2's strategy does not change: type S goes to C and type A goes to M.
2. If P1 stays at C it gets $¼ \cdot 2 + ¾ \cdot 0 = 0.5$; if it switches to M it gets $¼ \cdot 0 + ¾ \cdot 1 = 0.75$.

3. *0.75 > 0.5*: P1 deviates to M. The profile is not an equilibrium. Since *1/4 < 1/3*, by Table 4.16 there is no pure equilibrium; a mixed equilibrium is needed.

Example 4.26

Three-person TTC. The preferences are A: B ≻ C ≻ A, B: A ≻ C ≻ B, C: A ≻ B ≻ C. Who gets which room?

1. Round 1 arrows: A→B, B→A, C→A. A and B point to each other: A ↔ B is a cycle. A gets B's room and B gets A's room, and both leave.
2. Round 2: only C remains. Since A's and B's rooms are gone, C points to itself and keeps its own room.

C would most have liked A's room, but A does not prefer C's room to B's. C's room is nobody's first choice; C finds no one to trade with.

4.7 Summary

- **Mechanism design** is game theory in reverse: the desired outcome is given, and we look for rules (an allocation and a payment) that produce it in equilibrium. The difficulty is that the players' types (values, preferences, costs) are hidden from the designer.
- A **social choice function** f assigns the desired outcome to each type profile. In a **direct mechanism** the players report their types; the mechanism computes the outcome $x(b)$ and the payments $p(b)$; utility is quasi-linear: $u_i = v_i(x) - p_i$.
- **The revelation principle**: an outcome implemented in equilibrium by an indirect mechanism can also be implemented by a direct mechanism in which honest reporting is an equilibrium. If the equilibrium concept is dominant strategies we get DSIC; if it is Bayes–Nash we get BIC; DSIC ⇒ BIC.
- **IR** means that participating is no worse than staying out. IC and IR are different conditions. In the Ultimatum Game, theory gives the equilibria (10, 0) and (9, 1); in experiments, low offers are rejected: the utility model must include fairness.
- **Proportional cost sharing** is not IC: in the park example Cem raises his utility from 4 to 15 by reporting 5; if everyone reports half, the park is not built. **Myerson–Satterthwaite**: if values and costs are independent and come from overlapping ranges, efficiency, BIC, IR, and budget balance cannot all hold in bilateral trade.
- **Implementation theory**: honesty need not be the only equilibrium; in Vickrey, the bids (0, 100) form a Nash equilibrium that gives the item to the lower-valued player.

- **Myerson's lemma**: in a single-parameter environment, an allocation can be used in a DSIC mechanism if and only if it is monotone. The payment is uniquely determined: $p = b \cdot x(b) - \int_0^b x(z)\,dz$; for a 0/1 allocation it is the critical bid.
- **VCG** chooses the outcome with the highest social welfare and makes each player pay for their externality: p_i = (the others' best welfare without i) − (the others' welfare with i). It is DSIC, efficient, and (with the Clarke pivot) IR; it lacks strong budget balance. In the public project example the Gym is chosen and the payments are 0, 5, 20. For a single item, VCG = Vickrey.
- **In design without money**, Gibbard–Satterthwaite allows only dictatorship for general preferences and at least three alternatives, among rules in which every alternative can win. **Gale–Shapley** always finds a stable matching ($O(n^2)$), is best for the proposers and DSIC for them; the receiving side can gain by lying.
- The **median** is strategy-proof with single-peaked preferences and minimizes the total distance; the **mean** can be manipulated. The **Boston** mechanism is not DSIC; **serial dictatorship** and **TTC** are DSIC. TTC is Pareto optimal and in the core.
- **In a Bayesian game**, Nature chooses types according to a common prior; a strategy is a rule from types to actions; in a **Bayes–Nash equilibrium** every type maximizes its expected payoff. In the meeting game, for $P(S) = 1/2$ the only pure equilibrium is (C; S→C, A→M).

4.8 Key terms

mechanism design type social choice function direct mechanism quasi-linear utility
revelation principle incentive compatibility dominant-strategy incentive compatibility (DSIC)
Bayesian incentive compatibility (BIC) individual rationality free-rider social welfare
budget balance Myerson–Satterthwaite theorem implementation theory Myerson's lemma
critical bid VCG mechanism Clarke pivot rule externality Vickrey auction strategy-proof
Gibbard–Satterthwaite theorem stable matching blocking pair
Gale–Shapley algorithm (deferred acceptance) single-peaked preferences median mechanism
serial dictatorship Boston mechanism Top Trading Cycles (TTC) core Pareto optimal
Bayesian game common prior Bayes–Nash equilibrium

4.9 Test Yourself

1. A playground costs 40,000 TL. The values are Ayşe 20, Bora 16, Cem 14. The rule is proportional cost sharing: the playground is built if the reports sum to at least 40, and the cost is shared in proportion to the reports. What are the payments and utilities when everyone is honest? When

Bora and Cem are honest, what is Ayşe's most profitable report?

Answer

The total is $50 \geq 40$. The payments are $40 \cdot 20/50 = 16$, $40 \cdot 16/50 = 12.8$, $40 \cdot 14/50 = 11.2$; the utilities are 4; 3.2; 2.8 (IR holds). Bora's and Cem's reports sum to 30; for the playground to be built, Ayşe must report at least 10. If Ayşe reports 10, the total is 40 and she pays $40 \cdot 10/40 = 10$; her utility is $20 - 10 = 10 > 4$. The rule is not IC.

2. Three residents report the following values for three projects (Park, Library, Gym): Ayşe (20, 5, 10), Bora (0, 25, 15), Cem (15, 10, 30). Which project does VCG choose? What are each resident's payment and utility?

Answer

The totals are Park 35, Library 40, Gym 55: the Gym is chosen. Without Ayşe the totals are 15, 35, 45; the best is 45, and at the Gym the others get 45: payment 0. Without Bora, 35, 15, 40; the best is 40, and at the Gym the others get 40: payment 0. Without Cem, 20, 30, 25; the best is the Library (30), and at the Gym the others get $10 + 15 = 25$: payment $30 - 25 = 5$. The utilities are Ayşe $10 - 0 = 10$, Bora $15 - 0 = 15$, Cem $30 - 5 = 25$. The budget surplus is 5.

3. Four bidders for a single item have values 40, 65, 50, and 30. In a Vickrey auction, who wins and how much do they pay? How does the outcome change if the bidders with values 65 and 50 collude and the bidder with value 50 bids 0?

Answer

With honest bids, 65 wins and pays the second-highest bid, 50; the utility is 15. With collusion the bids are 40, 65, 0, 30. 65 still wins, but the second-highest bid is 40; the winner pays 40 and the utility rises to 25. The group gains 10 units, and the seller's revenue falls from 50 to 40. A deviation by one player alone would not pay, but collusion does.

4. In a single-parameter environment, with the others' bids fixed, a player's allocation is as follows: 0 below a bid of 3, 1/4 between 3 and 6, and 1 from 6 up. By the Myerson payment, what are $p(4)$ and $p(8)$? If the player with value 8 bids honestly, what is the utility?

Answer

The allocation is monotone. $\int_0^4 x = ¼ \cdot 1 = 0.25$, $p(4) = 4 \cdot ¼ - 0.25 = 0.75$ (check: jump × threshold = $¼ \cdot 3$). $\int_0^8 x = ¼ \cdot 3 + 1 \cdot 2 = 2.75$, $p(8) = 8 - 2.75 = 5.25$ (check: $¼ \cdot 3 + ¾ \cdot 6 = 0.75 + 4.5 = 5.25$). The utility is $8 - 5.25 = 2.75$, which is the area under the curve.

5. Student preferences are A: Y ≻ X ≻ Z, B: X ≻ Y ≻ Z, C: Y ≻ Z ≻ X; project preferences are X: A ≻ B ≻ C, Y: B ≻ C ≻ A, Z: C ≻ A ≻ B. When the students propose, how many proposals does Gale–Shapley

make and what is the result? What if the projects proposed?

Answer

The students propose: A→Y, B→X, C→Y; Y holds C and rejects A (C ≻ A). A→X; X holds A and rejects B. B→Y; Y holds B and rejects C. C→Z: accepted. With 6 proposals the result is A–X, B–Y, C–Z. If the projects propose, X→A, Y→B, Z→C and all three are accepted: the same result with 3 proposals. This matching is the only stable matching. Every student gets their second choice and every project gets its first choice.

6. Five villages are at km 1, 4, 6, 11, and 18. Where is the facility built under the mean and the median rules, and what is the total distance? Under the mean rule, what must the village at km 18 report to pull the facility exactly to itself?

Answer

Median 6: the total is $5 + 2 + 0 + 5 + 12 = 24$. Mean $40/5 = 8$: the total is $7 + 4 + 2 + 3 + 10 = 26$. For the mean to be 18, $(1 + 4 + 6 + 11 + x)/5 = 18$, that is, $x = 90 - 22 = 68$. Under the median, a report of 68 leaves the median at 6.

7. Four students' room preferences: A: B ≻ C ≻ A, B: C ≻ A ≻ B, C: B ≻ D ≻ C, D: A ≻ B ≻ D. Applying TTC, who gets which room?

Answer

Round 1 arrows: A→B, B→C, C→B, D→A. Starting from A, the path is A → B → C → B; the cycle is B ↔ C. B gets C's room and C gets B's. Round 2: A's first two choices are gone, so A points to itself; of D's first two choices, A remains: D→A. The cycle is A → A: A keeps its own room. Round 3: D points to itself and keeps its own room. The result is B ← C, C ← B, A ← A, D ← D.

8. In the meeting game, let the probability of type S be 2/5. Are (C; S→C, A→M) and (M; S→M, A→C) Bayes–Nash equilibria?

Answer

First candidate: C → $2/5 \cdot 2 = 4/5$, M → $3/5 \cdot 1 = 3/5$; C is better, so it is an equilibrium ✓. Second candidate: C → $3/5 \cdot 2 = 6/5$, M → $2/5 \cdot 1 = 2/5$; P1 deviates to C, so it is not an equilibrium. Since $1/3 \le 2/5 < 2/3$, there is a single pure equilibrium.

4.10 Further reading

- T. Roughgarden, *Twenty Lectures on Algorithmic Game Theory* (2016), Lectures 2–4, 7, and 10: gives the full proof of Myerson's lemma, VCG, kidney exchange, and stable matching in short lectures from a computer science viewpoint.
- N. Nisan, T. Roughgarden, É. Tardos, V. Vazirani (eds.), *Algorithmic Game Theory* (2007), Chapters 9 and 10: presents mechanism design with money and without money (median, housing market, stable matching) in detail and with proofs.
- D. Easley and J. Kleinberg, *Networks, Crowds, and Markets* (2010), Chapters 10 and 15: explains matching markets and the relation of VCG to sponsored search ads with little mathematics and intuitive examples.
- A. E. Roth, *Who Gets What — and Why* (2015): describes how real markets such as school placement, medical residency, and kidney exchange were designed, in the words of the researcher who designed them.

4.11 References

1. A. Abdulkadiroğlu, T. Sönmez, "School choice: A mechanism design approach", *American Economic Review* 93(3), 2003.
2. K. J. Arrow, "The property rights doctrine and demand revelation under incomplete information", M. Boskin (ed.), *Economics and Human Welfare*, Academic Press, 1979.
3. D. Black, "On the rationale of group decision-making", *Journal of Political Economy* 56(1), 1948.
4. C. F. Camerer, *Behavioral Game Theory*, Princeton University Press, 2003.
5. E. H. Clarke, "Multipart pricing of public goods", *Public Choice* 11, 1971.
6. C. d'Aspremont, L.-A. Gérard-Varet, "Incentives and incomplete information", *Journal of Public Economics* 11(1), 1979.
7. D. Easley, J. Kleinberg, *Networks, Crowds, and Markets*, Cambridge, 2010 (Chapters 10 and 15).
8. D. Gale, L. S. Shapley, "College admissions and the stability of marriage", *American Mathematical Monthly* 69(1), 1962.
9. A. Gibbard, "Manipulation of voting schemes: A general result", *Econometrica* 41(4), 1973.
10. T. Groves, "Incentives in teams", *Econometrica* 41(4), 1973.
11. W. Güth, R. Schmittberger, B. Schwarze, "An experimental analysis of ultimatum bargaining", *Journal of Economic Behavior and Organization* 3(4), 1982.
12. J. C. Harsanyi, "Games with incomplete information played by 'Bayesian' players, I–III", *Management Science* 14, 1967–68.
13. A. R. Karlin, Y. Peres, *Game Theory, Alive*, AMS, 2017 (mechanism design, stable matching).
14. E. Maskin, "Nash equilibrium and welfare optimality", *Review of Economic Studies* 66(1), 1999.
15. H. Moulin, "On strategy-proofness and single peakedness", *Public Choice* 35(4), 1980.

16. R. B. Myerson, “Optimal auction design”, *Mathematics of Operations Research* 6(1), 1981.

17. R. B. Myerson, M. A. Satterthwaite, “Efficient mechanisms for bilateral trading”, *Journal of Economic Theory* 29(2), 1983.

18. N. Nisan, T. Roughgarden, É. Tardos, V. Vazirani (eds.), *Algorithmic Game Theory*, Cambridge University Press, 2007.

19. M. J. Osborne, *An Introduction to Game Theory*, Oxford University Press, 2004 (Bayesian games).

20. A. E. Roth, “Incentive compatibility in a market with indivisible goods”, *Economics Letters* 9(2), 1982.

21. A. E. Roth, “The economics of matching: Stability and incentives”, *Mathematics of Operations Research* 7(4), 1982.

22. A. E. Roth, *Who Gets What — and Why*, Houghton Mifflin Harcourt, 2015.

23. A. E. Roth, A. Postlewaite, “Weak versus strong domination in a market with indivisible goods”, *Journal of Mathematical Economics* 4(2), 1977.

24. A. E. Roth, T. Sönmez, M. U. Ünver, “Kidney exchange”, *Quarterly Journal of Economics* 119(2), 2004.

25. T. Roughgarden, *Twenty Lectures on Algorithmic Game Theory*, Cambridge University Press, 2016.

26. M. A. Satterthwaite, “Strategy-proofness and Arrow's conditions”, *Journal of Economic Theory* 10(2), 1975.

27. L. Shapley, H. Scarf, “On cores and indivisibility”, *Journal of Mathematical Economics* 1(1), 1974.

28. Y. Shoham, K. Leyton-Brown, *Multiagent Systems*, Cambridge University Press, 2009.

29. W. Vickrey, “Counterspeculation, auctions, and competitive sealed tenders”, *Journal of Finance* 16(1), 1961.

Auctions

A seller does not know to whom to sell an item, or at what price, because the seller cannot see how much the buyers value it. An auction solves this problem with rules: bidders submit bids, and the rules determine the winner and the payment. But when the rules change, so does the behavior of a clever bidder. These notes begin by running the four basic auction formats on a small numerical example. They then prove why truthful bidding is best under the second-price rule, derive how much a bidder shades the bid under the first-price rule, and show why different rules bring the seller the same revenue on average. The last topics are the reserve price, the winner's curse, and two-sided markets.

LEARNING OUTCOMES

Running the formats
Find the winner and the payment in English, Dutch, first-price, and second-price auctions.

Proving truthfulness
Show by case analysis that truthful bidding is a weakly dominant strategy in the Vickrey auction.

Equilibrium bid
Derive the shading $b = (n-1)v/n$ under first price; compute revenue equivalence (2 bidders → 1/3) and the effect of a reserve price (5/12).

Market phenomena
Explain the winner's curse; find the matching and the price in a double auction.

5.1 Basic concepts

An **auction** is a method of sale in which one or more items are distributed according to the bids of the participants, under rules announced in advance. In this chapter the participants who want to buy the item are called **bidders**. The rules answer two questions: who gets the item, and what does that person pay? These two questions are exactly the **allocation rule** and the **payment rule** of mechanism design. Auctions are therefore the oldest and most widespread application of mechanism design.

A small game sets out the main question of the chapter. The place of auctions in everyday life, the basic terms, the probability tools needed for the examples, and three models of how bidders value an item frame this question.

5.1.1 A first game: the Vickrey auction

The rule of the game is short. Everyone writes a bid in a sealed envelope, and the envelopes are opened at the same time. The highest bidder gets the item. But the winner pays not their own bid but the *second-highest* bid. This rule is called the **Vickrey auction**, or the **second-price sealed-bid auction**, after the economist William Vickrey, who studied it in 1961 (Vickrey, 1961).

Three questions naturally arise for someone playing the game:

- If the item is worth 70 TL to a player, should the player write less than 70, exactly 70, or more in the envelope?
- Would the best bid change if the rivals' bids were known?
- If the winner paid their own bid (the first-price rule), would writing 70 still be a good idea?

Suppose that in one round the player writes 70, the first rival 60, and the second rival 45. The highest bid is the player's, and the player gets the item. The amount paid is the second-highest bid, that is, 60 TL. Since the item is worth 70 TL to the player, the player comes out ahead by 70 − 60 = 10 TL. Figure 5.1 shows this round.

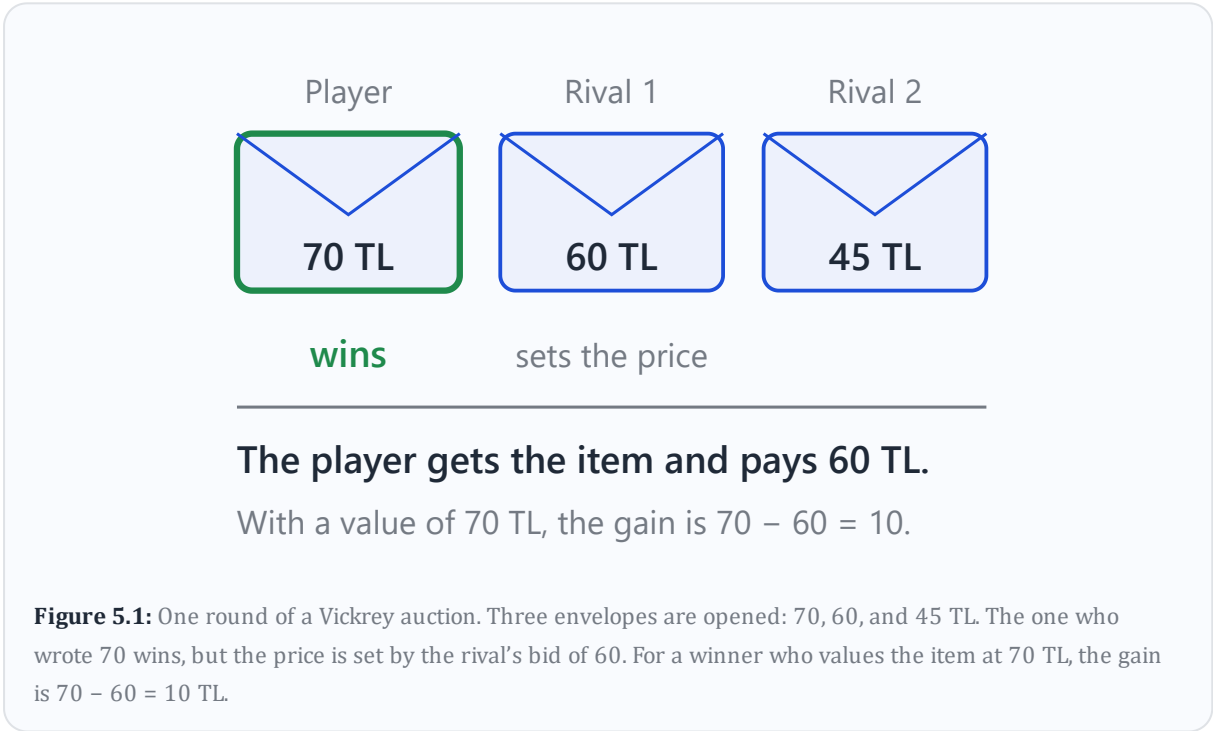


Figure 5.1: One round of a Vickrey auction. Three envelopes are opened: 70, 60, and 45 TL. The one who wrote 70 wins, but the price is set by the rival's bid of 60. For a winner who values the item at 70 TL, the gain is 70 − 60 = 10 TL.

The important detail in the figure is this: the 60 TL the winner pays is not the number written in the winner's own envelope. Had 65 or 100 been written instead of 70, the winner would not change and the amount paid would still be 60. The first big result of the chapter rests on this observation: under the second-price rule the bid determines only whether you win, not how much you pay. Why this observation makes the truthful bid the best strategy is proved step by step in 5.3.2. The answer to the third question is different: if the winner pays their own bid, a player who writes 70 and wins pays 70 and gains nothing. Under the first-price rule the bid is therefore shaded below the value; how far it is shaded is the subject of 5.3.5.

5.1.2 Why do auctions matter?

There are three reasons why auctions are so widespread.

First, they are **efficient in allocating resources**. In a well-designed auction the item tends to go to the person who values it most. This means that the total benefit society gets from the item is large. A painting going to the person who values it at 85 TL creates more welfare than its going to the person who values it at 55 TL.

Second, they provide **price discovery**. The seller does not know the buyers' values; if the seller did, the seller would sell directly to the highest-value buyer at that price. An auction pours the buyers' private information into bids, and the price emerges from these bids. Because the rules apply to everyone in the same way, the market becomes transparent.

Third, the auction is **the most widespread application of mechanism design**. A mechanism is a set of rules that takes information from participants (here, bids) and produces an outcome (who gets what, who pays what). An auction is the simplest combination of an allocation rule and a payment rule.

Area	How it is used
Works of art	The ascending-price English auction at auction houses.
Online advertising	Auctions run for ad slots on every search query.
Telecom licenses	Spectrum auctions; multi-item designs worth billions of lira.
Public tenders, treasury	Sealed-bid offers; treasury auctions.

Table 5.1: Main areas in which auctions are used.

The examples in Table 5.1 differ greatly in scale. At an art auction a single painting is sold; in a search engine thousands of small auctions run every second. In spectrum auctions hundreds of licenses are sold at the same time and bidders value the licenses jointly. In public tenders the direction is reversed: the state is the buyer and firms compete to offer the lowest price. The analyses in this chapter are set in the simplest case, with a single item and a single seller; but the ideas that come out of them (truthful bidding, shading, revenue equivalence, the reserve price) are also the foundation of the large applications.

5.1.3 Basic terms and the model

The analysis of auctions rests on four concepts. Telling them apart prevents most of the mistakes made throughout the chapter. The pair most often confused is *value* and *bid*.

Definition 5.1

- **Value** v_i: the true worth that bidder i places on the item; the highest amount the bidder is willing to pay for it. The value is *private information*: only the bidder knows it.
- **Bid** b_i: the number the bidder reports to the mechanism. The bid need not equal the value; the bidder can strategically write less or more.
- **Utility** u_i: if the bidder wins and pays p_i, $u_i = v_i - p_i$; if the bidder loses, $u_i = 0$.
- **Independent private values** (IPV): each bidder's value is drawn independently of the others', and each knows only their own value.

This definition of utility is called **quasi-linear utility**. The word "linear" refers to the payment being subtracted one for one from utility: paying 1 TL more reduces utility by exactly 1. A losing bidder does not get the item and pays nothing; the utility is zero. Losing is therefore not a "penalty"; nothing simply happens. The bidder's goal is to maximize utility.

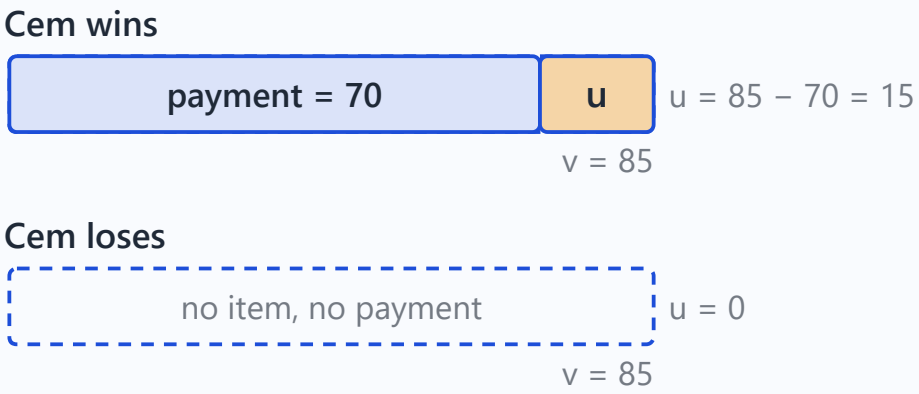


Figure 5.2: Quasi-linear utility. Cem's value is $v = 85$. Under the second-price rule he wins and pays the second-highest bid, 70: the 70 part of the value bar goes to the payment, and the remaining $u = 85 - 70 = 15$ is utility. Had he lost, there would be no item and no payment: $u = 0$.

Figure 5.2 shows utility as the splitting of a bar. The value is the whole bar; the payment "eats" part of the bar and the rest stays with the bidder. If the payment were larger than the value, the remainder would be negative; the bidder would lose out by getting the item. The danger of the "overbidding" strategy comes from this.

The IPV model has two parts. *Private* means that everyone knows their own value and that this value is not affected by what others know. Someone buying a painting for their own living room does not change their own value even on learning how much others like the painting. *Independent* means that values are generated as independent random numbers: knowing Ayşe's value says nothing about Bora's. IPV is the basic model of auction theory. Most of the theorems in this chapter hold in this model; when the model changes, the results may change too (5.1.5).

There are two more assumptions often used in the examples. The first is that values are **uniformly distributed** on the interval *[0, 1]*; this is written $v_i \sim U[0, 1]$. The second is that bidders are **risk-neutral**: a risk-neutral bidder compares uncertain outcomes by their expected utilities. For example, such a bidder sees an option that gives utility 10 or 0 with equal probability as equivalent to one that gives a sure utility of 5. These two assumptions are chosen to keep the calculations simple. The case of risk-averse bidders is examined separately in 5.4.3.

Common mistake

Taking value and bid to be the same thing. The value is the true number in the bidder's head and does not change; the bid is a decision made by looking at that value. A "truthful bid" means $b_i = v_i$; but under the first-price rule a rational bidder writes $b_i < v_i$. Utility is always computed from the *value*: $u_i = v_i - p_i$; the bid does not enter the utility formula directly, it only affects whether you win and, under some rules, the payment.

5.1.4 Probability tools used in the examples

The chapter's calculations rest on a few simple probability tools. They are collected briefly below.

The uniform distribution. $v \sim U[0, 1]$ means that v can take every value between 0 and 1 "with equal chance". More precisely, the probability of falling in any interval inside *[0, 1]* equals the length of that interval. For example, $P(v < 0.3) = 0.3$ and $P(0.2 < v < 0.7) = 0.5$. The function that gives this probability is called the **distribution function** (CDF) and is written $F(t) = P(v \le t)$; for $U[0, 1]$, $F(t) = t$. The derivative of the distribution function is the **density** $f(t)$; for $U[0, 1]$, $f(t) = 1$. The density shows how thickly probability is spread over the values; in the uniform distribution it is the same everywhere.

Products of independent events. If two events are independent, the probability that both occur is the product of the probabilities. If two bidders' values are independent $U[0, 1]$, the probability of the event "both values are below 0.5" is $0.5 \cdot 0.5 = 0.25$. All the winning-probability formulas in the chapter follow from this product rule: a bidder must beat *every* rival to win, and if the rivals are independent these probabilities multiply.

Expected value. The expected value of a random quantity is the average of the values it takes, weighted by their probabilities. The average obtained by repeating an experiment many times approaches the expected value. For $v \sim U[0, 1]$, $E[v] = 1/2$. Two kinds of expected value come up often in the chapter: a bidder's *expected utility* (what a bid earns on average) and the seller's *expected revenue* (what a rule brings the seller on average).

The tail formula. For a random X with values between 0 and 1, the expected value can also be computed with this formula:

$$E[X] = \int_0^1 P(X > t)\, dt$$

The meaning of the formula is as follows. Let X be the length of a bar starting at 0. At each point t, the probability that the bar extends past that point is $P(X > t)$. Adding up these probabilities over all t (taking the integral) gives the average length of the bar. For example, taking $X \sim U[0, 1]$ gives $P(X > t) = 1 - t$ and $\int_0^1 (1 - t)\, dt = 1 - 1/2 = 1/2$; this is the familiar $E[X] = 1/2$. The tail formula is very useful for finding the expectation of the maximum or minimum of several numbers, because the probability of events such as "the largest is below t" is easily written with the product rule (5.4.1).

Integrals of polynomials. The chapter needs only integrals of polynomials: $\int_0^a t^k\, dt = a^{k+1}/(k + 1)$. For example, $\int_0^1 t^2\, dt = 1/3$ and $\int_0^r 2t^2\, dt = 2r^3/3$.

Derivatives and maxima. A bidder's expected utility is a function of the bid. If this function first rises and then falls, it takes its largest value where the derivative is zero. Two rules are used: the derivative of t^k is $k\, t^{k-1}$, and the derivative of a product of two functions is $(g\, h)' = g'\, h + g\, h'$ (the product rule).

Example 5.1

The values of three bidders are independent $U[0, 1]$. What is the probability that a bidder whose value is 0.6 has a value higher than both of the other two bidders?

1. For a single rival, $P(v_j < 0.6) = F(0.6) = 0.6$ (in the uniform distribution the probability is the length of the interval).
2. The two rivals' values are independent; the probability that both are below 0.6 is the product: $0.6 \cdot 0.6 = 0.36$.
3. Result: the bidder's value is the highest with probability 36%. In general, with n bidders this probability is v^{n-1}.

5.1.5 Valuation models

How bidders value the item determines how the auction is analyzed. Three basic models are distinguished in the literature.

Definition 5.2

- **Private value**: each bidder determines the item's value only from their own information and taste. An example is a painting to hang in one's own home. The values differ: $v_1 \neq v_2 \neq \ldots$
- **Common value**: the item's true value is the same for everyone, but no one knows it exactly; the participants' estimates differ. An example is the market value of the oil in an oil field.
- **Interdependent values**: a bidder's value depends both on their own private information and on the others' information: $v_i = f(\text{own signal, other signals})$.

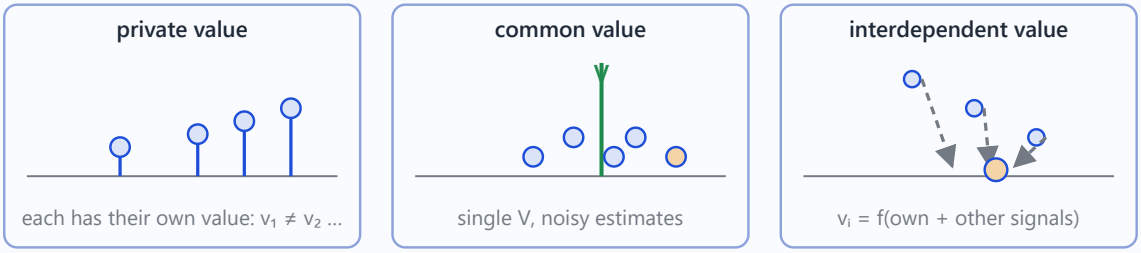


Figure 5.3: Three valuation models. On the left, private value: each bidder has their own value and the values differ. In the middle, common value: noisy estimates around a single true value V (green line). On the right, interdependent values: a bidder's value is affected by the others' signals as well as their own.

Figure 5.3 visualizes the difference among the three models. In the private value model, learning what others think changes no one's value. In the common value model the situation is reversed: a firm that learns another firm rated the field low suspects that its own estimate may be too optimistic. Information is valuable because everyone's estimate is a noisy measurement of the same truth.

Most real items lie somewhere between the two extremes. A work of art is valued both by personal taste (the private component) and by the price at which it can be sold later (the common component). In a corporate acquisition, the buyer's own synergies are the private component and the target's cash flows the common one. The interdependent values model represents this mixed case.

The difference among the models is not only theoretical. The common value model leads to a phenomenon called the **winner's curse**: the winner of the auction is most likely the participant with the most optimistic estimate of the value, and if this is not taken into account the winner loses on average. This phenomenon is examined numerically in 5.4.8. For the rest of the chapter the IPV model is assumed unless stated otherwise.

5.2 The four basic formats

Auction formats are classified by two criteria. The first is how bids are made: in front of everyone while the price changes (an open, “outcry” auction) or once and secretly (sealed-bid). The second is what the winner pays: their own bid or the second-highest bid. Combining the two criteria gives the four basic formats. Running all four on the same four-person example shows the differences clearly.

5.2.1 The four rules

Format	How it works	Winner	Pays
English	The price rises openly; bidders drop out one by one.	The last one remaining	The price at which the last rival dropped out
Dutch	The price starts high and falls; the first to say “accept” buys.	The first to accept	The price at that moment
First-price sealed-bid	Bids are submitted secretly, once.	The highest bid	Their own bid
Second-price (Vickrey)	Bids are submitted secretly, once.	The highest bid	The second-highest bid

Table 5.2: The four basic auction formats.

The **English (ascending) auction** is the classic format seen at auction houses. The auctioneer starts at a low price and raises it step by step. When the price passes the amount a bidder is willing to pay, that bidder drops out. When one person remains, the auction ends; the last one remaining buys the item at the price at which the last rival dropped out. In theoretical analysis the “clock” version is usually used: the price rises continuously and bidders drop out by releasing a button. A bidder’s only decision is then reduced to the question “at what price will I drop out?”

The **Dutch (descending) auction** takes its name from the flower auctions in the Netherlands. The price starts at a high value and falls. The first to say “accept” buys the item at the current price, and the auction ends immediately. Because it is fast, it is used in flower and fish markets where many small lots are sold.

The **first-price sealed-bid auction** is the familiar form of public tenders. Each bidder submits a bid once and secretly. The highest bid wins and the winner pays their own bid.

The **second-price (Vickrey) auction** is also secret and one-shot. The winner is again the highest bidder, but the amount paid is the second-highest bid.

To compare the four formats the same four people are used. The highest amount each person is willing to pay for the item, that is, their *limit*, is: **Ayşe 70, Bora 55, Cem 85, Deniz 60**. These limits are private values. For now we assume that everyone bids their limit, that is, stays in an English auction up to their limit and accepts in a Dutch auction when the price comes down to their limit. When this assumption is sensible is examined in 5.3.

5.2.2 The English auction, step by step

Let the price start at 40 and rise slowly. Each bidder stays in as long as the price is below their own limit.

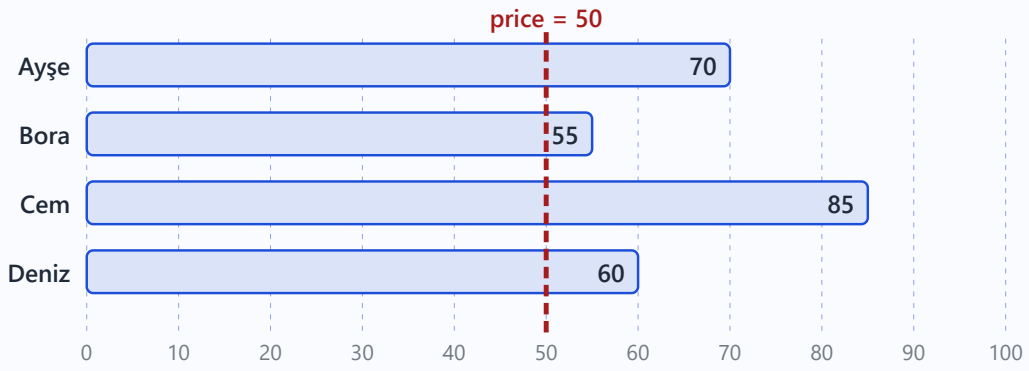


Figure 5.4: The start of the English auction. The bars are the four bidders' limits; the red dashed line is the current price. At price = 50, all limits are above 50 and all four are still in.

Example 5.2

1. **Price 50:** the smallest limit is 55; everyone is in (Figure 5.4).
2. **When the price passes 55**, Bora drops out, because buying at this price would cost him (his limit is 55). Remaining: Ayşe, Cem, Deniz.
3. **When the price passes 60**, Deniz drops out (her limit is 60). Remaining: Ayşe, Cem.
4. **When the price passes 70**, the last rival, Ayşe, drops out. Cem is left alone and wins.
5. **Payment:** Cem pays the price at which Ayşe dropped out, that is, about 70. If the price rises in steps of 1 TL, Cem pays 70 or 71; as the increment shrinks the payment approaches exactly 70. Cem's utility is about *85 − 70 = 15*.

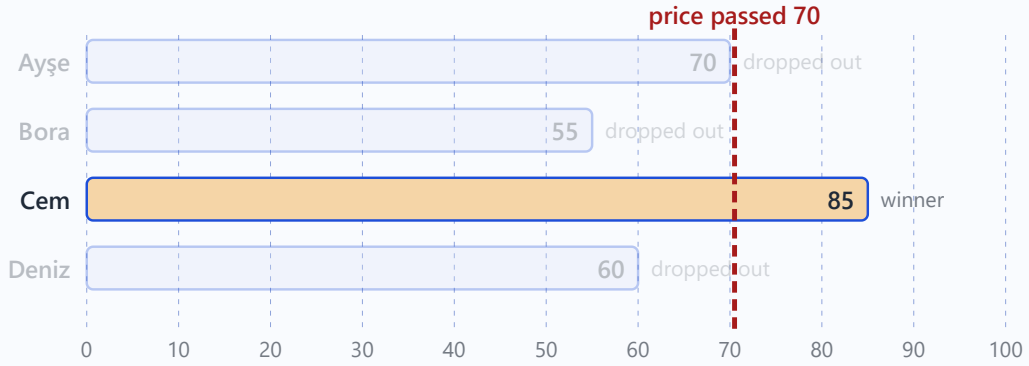


Figure 5.5: The end of the English auction. When the price passes 70, Ayşe drops out; the faded bars are those who dropped out. Cem wins and pays about 70, the second-highest value.

The outcome seen in Figure 5.5 is a general property of the English auction: *the winner is the person with the highest value, and the price paid is the second-highest value*. The reason is simple. The auction continues until the person with the second-highest value drops out; when that person drops out no rival remains. Cem's own limit of 85 has no effect on the price he pays. Had it been 75 or 100 instead of 85, Cem would still have won at 70. This is the same as the Vickrey observation in 5.1.1.

With private values, a bidder's best strategy in an English auction is to stay in up to their limit. Dropping out before the limit may forgo a profit that could have been made: Ayşe, dropping out at 65, could have bought the item at a price just over 60 had Cem not been there. Staying in beyond the limit, on the other hand, risks winning at a loss. This reasoning is the English-auction counterpart of the truthfulness proof for the Vickrey auction (5.3.2).

5.2.3 The Dutch auction, step by step

In a Dutch auction the price starts at 100 and falls by 5 at each step. Each bidder says "accept" when the price comes down to their limit.

Example 5.3

1. **Price 100:** this price is above everyone's limit; no one accepts, and the price falls by 5.
2. **Price 95:** the highest limit is 85; still no one accepts.
3. **Price 90:** again no one accepts.
4. **Price 85:** the price reaches Cem's limit. Cem says "accept" and pays 85. The auction ends at this moment; the others never get to speak.
5. **Result:** Cem wins and pays 85; his utility is *85 − 85 = 0*.

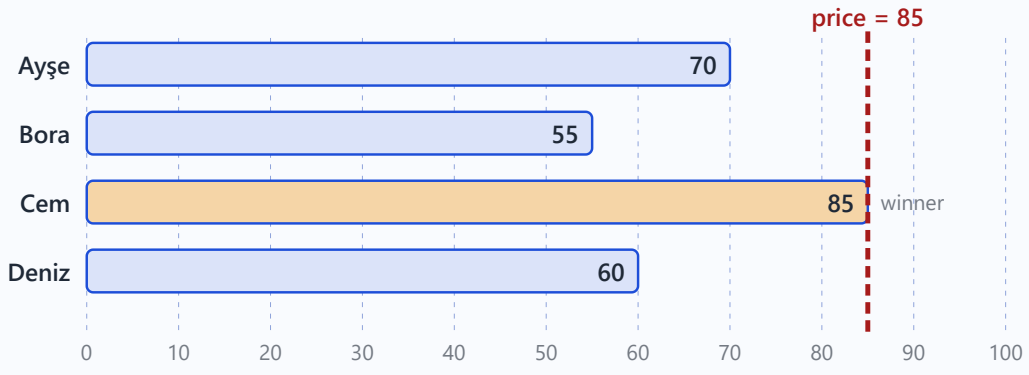


Figure 5.6: The end of the Dutch auction. The price falls from 100 in steps of 5; when it reaches 85 it hits Cem's limit, and Cem accepts and pays 85.

Comparing Figure 5.6 with the English auction figures shows an important difference. In the English auction, information emerges during the process: everyone sees who drops out at which price. In the Dutch auction, because the first to accept wins, the auction ends and no one learns anything about

anyone else's limit. A bidder answers only a single question in advance: "at what price will I accept?" The answer is a single number, and in effect this number is a hidden bid. The consequences of this observation are discussed in 5.2.5.

In the example Cem's utility came out zero because he accepted at his limit. This is not realistic behavior. Had Cem waited a little instead of accepting at 85, for example accepting at 75, he would still have won as long as the rivals' limits were below 75, and would have gained 10. The risk of waiting is that when the price falls to a rival's limit, that rival accepts first. This trade-off is the same as bid shading in the first-price auction.

5.2.4 First-price and second-price sealed-bid

In a sealed-bid auction everyone submits a bid once and secretly. No one sees the others' bids; the envelopes are opened at the same time. If the four people write their limits, the envelopes contain 70, 55, 85, and 60.

Example 5.4

1. **The envelopes are opened:** Ayşe 70, Bora 55, Cem 85, Deniz 60. Under both rules the winner is Cem, who made the highest bid.
2. **First price:** Cem pays his own bid, 85. Since his value is 85, his utility is $85 - 85 = 0$.
3. **Second price (Vickrey):** Cem pays the second-highest bid, Ayşe's 70. His utility is $85 - 70 = 15$.

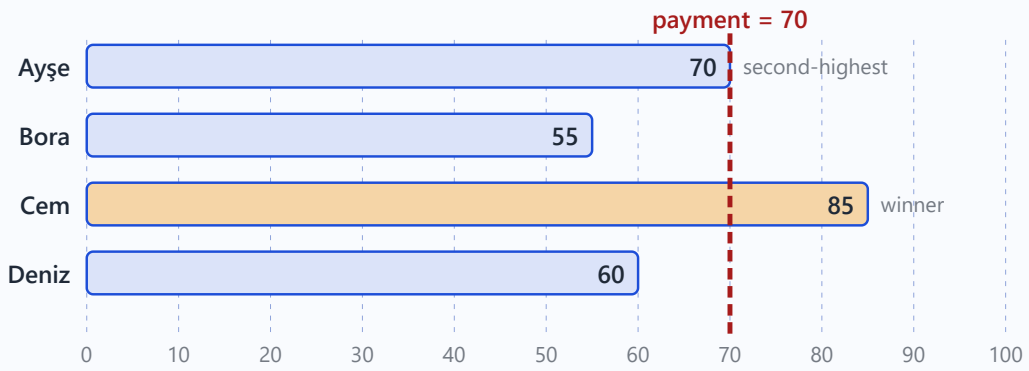


Figure 5.7: Second-price sealed-bid. The envelopes open as 70, 55, 85, 60; the winner is Cem (orange) and the payment is set by the second-highest bid, 70 (red dashed line).

Under the first-price rule the winner's payment equals their own bid; under the second-price rule it is independent of their own bid. In Figure 5.7, however long Cem's bar is, the payment line stays at 70. This single difference completely changes the behavior of a clever bidder under the two rules: under second price, writing your value is best; under first price, you must go below your value.

5.2.5 The same bids, four outcomes

The outcomes of the four rules are set side by side in Table 5.3.

Format	Winner	Payment	Whose bid sets the payment?
English	Cem	≈ 70	The last rival to drop out (Ayşe)
Second price	Cem	70	The second-highest bid (Ayşe)
Dutch	Cem	85	The winner
First price	Cem	85	The winner

Table 5.3: The outcome of the four rules when the limits are 70, 55, 85, 60 and everyone bids their limit.

The table reveals two pairs.

English ≈ second price. With private values, in both formats the winner is the person with the highest value, and pays the second-highest value. In the English auction the reason is that staying in up to your limit is dominant; in second price, that the truthful bid is dominant. The "≈" sign between the two formats stands for the small difference created by the price increment: in the English auction the payment is within one increment of the second-highest value. This equivalence is also exact only for private values. With common or interdependent values, seeing at which prices the rivals drop out in an English auction gives information; this information is absent in a sealed-bid auction, and the two formats can diverge.

Dutch ≡ first price. These two formats are *strategically equivalent*, regardless of the valuation model. In a Dutch auction a bidder gets no useful information until the auction ends, because the auction ends the moment someone accepts. Fixing in advance a number such as "I will accept when the price falls to this value" is therefore the same as writing this number in a sealed envelope. In both formats whoever has the highest number wins and pays their own number. The triple-bar sign (≡) stands for this identity being exact rather than approximate.

Common mistake

Looking at Table 5.3 and concluding that "first price earns the seller more, because 85 > 70". In the table everyone bids their limit; this assumption is not rational under the first-price rule. A rational bidder shades the bid under first price, that is, bids below the value. Once shading is taken into account, the *expected* revenues the two rules bring the seller become equal. This result is called **revenue equivalence** and is proved in 5.4.2.

Example 5.5

Let the limits be 40, 65, 80, and 90, and let everyone bid their limit. Who wins under the four rules and what do they pay?

1. Under all four rules the winner is the bidder with the highest limit, 90.
2. **English:** when the price passes 40 and 65 two bidders drop out; when it passes 80 the last rival drops out. Payment ≈ 80, utility ≈ 10.
3. **Second price:** the envelopes are 40, 65, 80, 90; the payment is the second-highest bid, 80; utility 10.
4. **Dutch:** when the price falls to 90 the winner accepts; payment 90, utility 0.
5. **First price:** the winner pays their own bid, 90; utility 0.

5.3 Sealed-bid strategies

In the earlier examples everyone was assumed to bid their limit. Once this assumption is dropped, the question is: what should a clever bidder write in the envelope? The answer differs greatly by rule. Under the second-price rule the answer is "your value", and it does not depend on what the rivals do. Under the first-price rule the answer depends on the rivals' behavior and requires an equilibrium computation.

5.3.1 The strategic property of the Vickrey auction

The rule of the Vickrey auction is this: the highest bidder wins and pays the second-highest bid. The most important consequence of this rule is that the bidder need not guess the rivals. To state this we first recall the concept of a dominant strategy.

Definition 5.3

A strategy is a **weakly dominant strategy** if, against *every* possible choice of the others, it does as well as any other strategy and against at least one choice it does strictly better. In an auction this means that the bidder's utility is, for every bid profile of the rivals, at least as large as with any other bid and for some profiles strictly larger.

Theorem 5.1 (Vickrey, 1961)

In the single-item model with independent private values, in a second-price sealed-bid auction, bidding the true value ($b_i = v_i$) is a weakly dominant strategy for every bidder.

The idea behind the theorem can be put in one sentence: *the bid determines only whether you win, not what you pay*. If the bidder wins, they pay the rivals' highest bid; this amount is independent of the number in their own envelope. The only job of the number written in the envelope is therefore to set

a threshold: if the rivals' highest bid is below this threshold the bidder wins, if above, the bidder loses. The best threshold is the point that separates where winning is profitable from where it is costly; and that point is the bidder's value itself.

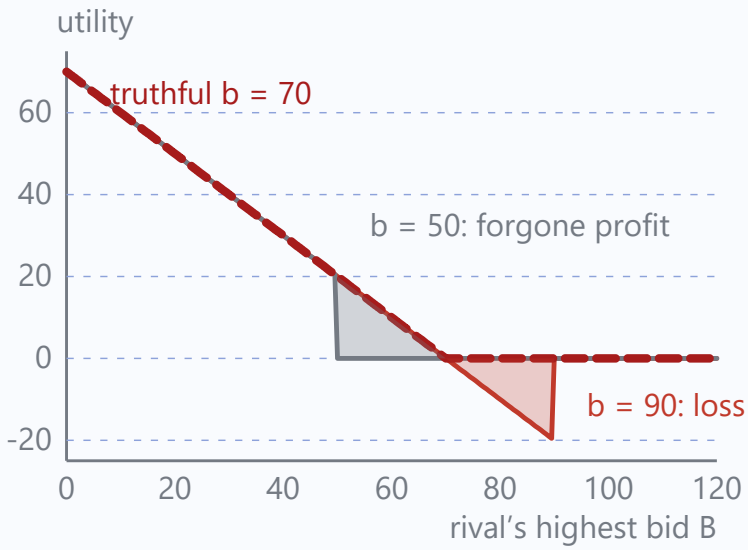


Figure 5.8: How utility varies with the rivals' highest bid B when the value is $v = 70$. The dashed red curve is the utility of the truthful bid $b = 70$: $70 - B$ for $B < 70$, then 0. The gray region shows the profit forgone by the bid $b = 50$, and the red region the loss suffered by the bid $b = 90$. No bid can exceed the truthful bid's curve for any B.

Figure 5.8 is a visual summary of the proof. The horizontal axis is the rivals' highest bid B. When B is below 70, the truthful bidder wins and gets utility $70 - B$; this utility falls linearly as B grows. When B passes 70, the truthful bidder loses and the utility is 0. A bidder who bids 50 loses when B is between 50 and 70, although winning would have been profitable in this range (gray region). A bidder who bids 90 wins when B is between 70 and 90, but winning is costly in this range (red region, negative utility). Each deviation does the same as the truthful bid at some values of B and worse at others; it is nowhere better.

The conditions of validity of the theorem matter. The result is proved in the single-item model with independent private values. It may not hold in the same form with common values, with budget constraints, or when values are interdependent. For example, a bidder whose budget is smaller than their value cannot bid their value; with common values the "value" is not even known. The Vickrey auction is the single-item special case of the VCG (Vickrey–Clarke–Groves) mechanism of mechanism design. In VCG too, each participant pays the harm they impose on the others; for a single item this harm is the second-highest value.

5.3.2 Proof of truthfulness, step by step

The proof is a *case analysis*: the possible cases are examined one by one for the rivals' highest bid $B = max_{j \neq i} b_j$. The bidder does not know B; but since the proof is carried out separately for every B, this ignorance is no problem. We begin with a numerical example: the value is $v = 70$ and three strategies are compared: truthful $b = 70$, over $b = 90$, under $b = 50$.

Example 5.6

1. **Truthful, B = 60:** since 70 > 60, the bidder wins and pays 60; utility $70 - 60 = 10$.
2. **Truthful, B = 80:** since 70 < 80, the bidder loses; utility 0. The truthful bid always makes the right decision: it wins when there is a profit and loses when there would be a loss.
3. **Over b = 90, B = 60:** $B < v$. It wins again and pays 60 again; utility 10. No difference from truthful.
4. **Over b = 90, B = 80:** B is between v and b. It wins but pays 80; utility $70 - 80 = -10$. Truthful would have got 0: a loss.
5. **Over b = 90, B = 95:** $B > b$. It loses; utility 0, same as truthful.
6. **Under b = 50, B = 40:** $B < b$. It wins and pays 40; utility 30. The truthful bid would also have paid 40; no difference, because B, not b, sets the payment.
7. **Under b = 50, B = 60:** B is between b and v. It loses; utility 0. Truthful would have gained $70 - 60 = 10$: an opportunity missed.
8. **Under b = 50, B = 80:** it loses; same as truthful.

Strategy	B (rival)	Outcome	Utility	Relative to truthful
Truthful b = 70	60	wins, pays 60	10	—
Truthful b = 70	80	loses	0	—
Over b = 90	60	wins, pays 60	10	same
Over b = 90	80	wins, pays 80	−10	worse
Over b = 90	95	loses	0	same
Under b = 50	40	wins, pays 40	30	same
Under b = 50	60	loses	0	worse
Under b = 50	80	loses	0	same

Table 5.4: Case analysis in Vickrey for value $v = 70$. The last column gives the deviation relative to the truthful bid.



Figure 5.9: The number line of the last case: the under-bid $b = 50$, value $v = 70$, and rival bid $B = 80$. Since B is to the right of both, both strategies lose; the outcome is the same.

Table 5.4 shows a pattern. The outcome of a deviation depends on whether B falls between b and v. If B is outside this interval, the deviating bidder makes the same decision as the truthful bidder (both win or both lose), and if winning, pays the same B. If B is inside the interval, the decision changes, and the changed decision is always in the wrong direction. Figure 5.9 shows this interval on a number line: the region between b and v is the "danger zone". As a numerical check, when all pairs $b, B \in \{0, 1, \ldots, 120\}$ are tried for $v = 70$, no b beats the truthful bid.

The same reasoning is written in general as follows.

Proof. Let the bidder's value be v and the rivals' highest bid B. With any bid, two outcomes are possible: winning (utility $v - B$, because the winner pays B) or losing (utility 0). So, with B fixed, whatever bid is made the utility is either $v - B$ or 0. The larger of these two numbers is $\max(v - B, 0)$, and no bid can give more than this.

The truthful bid gives exactly this maximum. If $B < v$, the truthful bidder wins and gets $v - B > 0$. If $B > v$, the bidder loses and gets 0; in this case $v - B < 0$, so 0 is better. If $B = v$, both outcomes give utility 0 and how the tie is broken is immaterial. So for every B the truthful bid gives the largest attainable utility; no other bid can be better than it.

For strictness it suffices to show that every deviation is strictly worse somewhere. If $b > v$, then for a B with $v < B < b$ the deviating bidder wins and gets $v - B < 0$; the truthful bid would have given 0. If $b < v$, then for a B with $b < B < v$ the deviating bidder loses and gets 0; the truthful bid would have given $v - B > 0$. Thus $b = v$ is a weakly dominant strategy. □

The proof never used how the rivals bid. The rivals may be truthful, may lie, may act at random; the result does not change. This is why dominant strategy is a stronger concept than Nash equilibrium: in a Nash equilibrium a player's strategy is best against the rivals' equilibrium strategies, whereas a dominant strategy is best against *every* strategy of the rivals. Mechanisms with this property are called **strategy-proof** or dominant-strategy incentive compatible (DSIC).

Common mistake

Thinking that "a truthful bid guarantees winning" or that "a truthful bid always gives positive utility". A truthful bid loses when $B > v$ and gets utility 0. Weak dominance says that the truthful bid is *at least as good as any other bid* in every case; it does not say that it pays off in every case. Another common mistake is to think that dominance holds "if the rivals are

truthful too"; the proof is independent of the rivals' behavior.

5.3.3 Back to the game: theory and lab

The theoretical answer to the opening question in 5.1.1 is now clear: write your value in the envelope. In the opening envelopes the player wrote 70 and the rivals 60 and 45. With the truthful bid the player wins and pays 60; the gain is 10. Had the player written 65 or 100, the player would still win and pay the same 60. Had the player written 55, the item would go to the rival who wrote 60 and the player would miss the gain of 10. When the bid is moved up or down from the value, the gain does not increase for any B; it only decreases for some values of B.

The answer to the third opening question is also clear. Under the first-price rule, a player who writes 70 gets a gain of $70 - 70 = 0$ even if winning. There the bid must be shaded below the value; how to shade it is the subject of 5.3.4.

Real behavior may differ from theory. In laboratory experiments it is common to bid *above the value* in second-price sealed-bid auctions, and this does not disappear easily even when the experiment is repeated. The same subjects approach equilibrium much faster in the ascending-price English auction (Kagel, Harstad, and Levin, 1987). One possible reason is this: the cost of overbidding appears only when B falls in a certain range, and this rarely happens; because subjects seldom see the loss, they do not change their habits. In an English auction, on the other hand, the moment the price passes the value is in plain view and dropping out is a natural decision.

A very close everyday example of the second-price rule is eBay's automatic (proxy) bidding system. The user tells the system the highest amount they are willing to pay; on the user's behalf the system bids the smallest amount that beats the rivals. The winner ends up paying a price one increment above the second-highest declared amount. This is an incremental version of the second-price rule.

5.3.4 Bid shading under first price

Under the first-price rule a truthful bid is pointless: even if the bidder wins, the utility is $v - v = 0$. Moving the bid below the value is called **bid shading**. Shading creates a trade-off. Lowering the bid increases the profit left on winning ($v - b$) but reduces the probability of winning. A bid's expected utility is the product of these two effects:

$$E[u(b)] = (v - b) \cdot P(\text{winning with } b)$$

The first factor in the formula is the profit on winning, the second the probability of that profit. Since the utility is 0 on losing, no other term enters the formula. At $b = 0$ the probability of winning is 0; at $b = v$ the profit is 0. At both ends the expected utility is zero; the best bid lies somewhere in between.

The probability of winning depends on how the rivals bid. So under first price the best bid, unlike under second price, is not independent of the rivals. When each bidder's strategy depends on the others', the right solution concept is an equilibrium. Here, however, each bidder does not know the

rivals' values, only their distribution. The equilibrium concept used in games of incomplete information of this kind is Bayes–Nash equilibrium.

Definition 5.4

Each bidder's **strategy** is a function from value to bid: $b_i = \beta_i(v_i)$. A strategy profile is a **Bayes–Nash equilibrium** if, when the others play their strategies and their values are random according to their distributions, $\beta_i(v_i)$ maximizes expected utility for every bidder i and every value v_i. If everyone uses the same function, the equilibrium is **symmetric**.

The definition carries the idea of Nash equilibrium, "no one wants to deviate alone", over to incomplete information. The difference is that the deviation is evaluated through expected utility: because the bidder does not know the rivals' values, the bidder considers the average outcome of each bid.

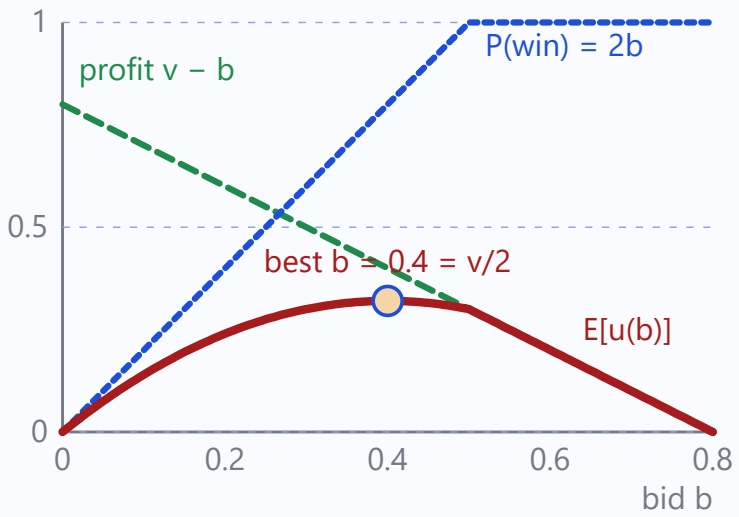


Figure 5.10: Two bidders, value $v = 0.8$; the rival bids half their value. The green dashed line is the profit left on winning, $v - b$; the blue dashed line is the winning probability $P(win) = 2b$ (1 for $b \geq 0.5$); the red curve is their product, the expected utility. The peak of the curve is at $b = 0.4 = v/2$.

Figure 5.10 makes the trade-off concrete. The rival's value is $U[0, 1]$ and the rival bids half of it; that is, the rival's bid is uniformly distributed between 0 and 0.5. To win with bid b, the rival's bid must be below b: $P(v_r/2 < b) = P(v_r < 2b) = 2b$ (for $b \leq 0.5$). The expected utility is then $(0.8 - b) \cdot 2b$. This is a piece of a parabola; its derivative $2(0.8 - b) - 2b = 1.6 - 4b$ is zero at $b = 0.4$. At the peak the expected utility is $(0.8 - 0.4) \cdot 0.8 = 0.32$.

The Dutch auction is strategically the same as this analysis. The answer to the question "at what price will I accept?" is a shaded bid. A bidder with value 0.8 accepts when the price falls to 0.4, when the rival behaves in the same way.

5.3.5 Deriving the equilibrium bid

In the general case there are n bidders and the values are independent $U[0, 1]$. We look for a symmetric equilibrium. To do so we assume a guessed form, compute the best response, and show that the result is consistent with the assumption. The assumption is this: the rivals bid a fixed fraction of their values.

Example 5.7

1. **Assumption.** Let the other $n - 1$ bidders bid $b_j = \beta \cdot v_j$, where β is a constant between 0 and 1 and $v_j \sim U[0, 1]$.
2. **Winning probability.** To win with bid b, every rival's bid must be below b. For one rival, $P(\beta\, v_j < b) = P(v_j < b/\beta) = b/\beta$ (for $b \le \beta$; in the uniform distribution the probability is the length of the interval). Since the rivals are independent, the probabilities multiply: $P(win) = (b/\beta)^{n-1}$.
3. **Expected utility.** Profit on winning times the probability of winning: $E[u(b)] = (v - b) \cdot (b/\beta)^{n-1}$.
4. **Derivative.** The derivative of $(v - b)$ is -1, and that of b^{n-1} is $(n - 1)\, b^{n-2}$. Combining with the product rule and factoring out the common factor b^{n-2}/β^{n-1} gives: $dE[u(b)]/db = [\, -b + (n - 1)(v - b)\,] \cdot b^{n-2}/\beta^{n-1}$. At the maximum the derivative is zero.
5. **Solution.** The outside factor is positive when $b > 0$; so the bracket must be zero: $-b + (n - 1)(v - b) = 0$. Expanding gives $-b + (n - 1)v - (n - 1)b = 0$, that is, $n\, b = (n - 1)v$ and $b = (n - 1)v/n$. After simplification the solution comes out independent of β.
6. **Consistency.** The best response found is "bid $(n - 1)/n$ times your value". This has the same form as the assumed one, and if $\beta = (n - 1)/n$ is chosen, everyone is giving the best response to the rivals: a symmetric Bayes–Nash equilibrium. For $n = 2$, $b = v/2$: half the value with two bidders.

That the solution found in the fifth step is a maximum can also be checked from the sign of the derivative. The bracket equals $(n - 1)v - n\, b$. It is positive when b is below the equilibrium bid and negative when above. So the expected utility first rises, peaks at the equilibrium bid, and then falls. The point found is indeed the maximum. Also, bids $b > \beta$ always win, but since at $b = \beta$ the bidder already wins with probability 1, these bids simply mean paying more; the best bid does not fall in this region.

The expected utility of a bidder with value v in equilibrium can also be computed. While the rivals bid $\beta\, v_j$, the bidder bids $\beta\, v$; this bid wins only if every rival's value is below v. The winning probability is v^{n-1} and the profit is $v - (n - 1)v/n = v/n$. The product is v^n/n. For $n = 2$, $v = 0.8$ this gives $0.64/2 = 0.32$, the same as the peak value in Figure 5.10.

The formula can also be checked numerically. For $n = 2, 3, 5, 10$ and $v = 0.3; 0.6; 0.8$, when $E[u(b)]$ is maximized on a grid of 10,001 points, the best bid coincides with $(n - 1)v/n$ in every case.

Common mistake

Taking the equilibrium bid to be a rule independent of the rivals' behavior. $b = (n-1)v/n$ is best when the values are $U[0, 1]$ and the rivals bid in proportion to their values (in the form βv); β itself does not change the result (as long as the candidate bid does not exceed β). For example, if the single rival ($n = 2$) bid truthfully, the winning probability would be $P(v_r < b) = b$ and the best response maximizing $(v - b) \cdot b$ would again be $b = v/2$; but when the distribution changes, so does the formula. The truthful bid in Vickrey, on the other hand, is best in every case; the two results are not equally strong.

5.3.6 Code trace: best_bid()

The result of the derivation can be compared with the best bid found by a computer program by brute force. The function below computes the expected utility of a bidder with value v, while the rivals bid βv, for every bid on the grid $0, step, 2 \cdot step, \ldots$ and returns the largest.

```
def best_bid(v, n, step=0.1):
  beta = (n - 1) / n        # rivals bid beta*v
  best, b_best = -1, 0
  b = 0.0
  while b <= v:
    p = min(1, b / beta) ** (n - 1)  # P(win)
    eu = (v - b) * p
    if eu > best:
      best, b_best = eu, b
    b = round(b + step, 10)
  return b_best, best
```

The code works line by line as follows:

- `beta = (n - 1) / n` : the fraction of their values that the rivals bid. For $n = 2$ it is 0.5.
- `best, b_best = -1, 0` : the largest expected utility so far and the bid that gives it. Initially `best` is −1; since expected utility is never negative, the first bid certainly becomes the "new best".
- `while b <= v` : bids are tried from 0 up to the value. Bidding above the value is pointless because it gives negative utility on winning.
- `p = min(1, b / beta) ** (n - 1)` : the winning probability $(b/\beta)^{n-1}$. The guard `min(1, …)` prevents the ratio from exceeding 1 when $b > \beta$; a probability cannot be larger than 1.
- `eu = (v - b) * p` : the expected utility. If it is larger, the best is updated.
- `b = round(b + step, 10)` : the next bid. `round` removes the small errors that arise because 0.1 cannot be represented exactly in binary (e.g. 0.30000000000000004); otherwise at the last step `b` could slightly exceed 0.8 and the loop could end early.

In the call `best_bid(0.8, 2)` , $\beta = 0.5$ and the grid is $b = 0; 0.1; \ldots; 0.8$. Table 5.5 shows each pass.

b	P(win)	E[u] = (0.8 − b) · P	Comparison	(best, b_best)
0	0	0.8 · 0 = 0	0 > −1: new best	(0; 0)
0.1	0.2	0.7 · 0.2 = 0.14	new best	(0.14; 0.1)
0.2	0.4	0.6 · 0.4 = 0.24	new best	(0.24; 0.2)
0.3	0.6	0.5 · 0.6 = 0.30	new best	(0.30; 0.3)
0.4	0.8	0.4 · 0.8 = 0.32	new best	(0.32; 0.4)
0.5	1	0.3 · 1 = 0.30	0.30 < 0.32: unchanged	(0.32; 0.4)
0.6	1	0.2 · 1 = 0.20	unchanged	(0.32; 0.4)
0.7	1	0.1 · 1 = 0.10	unchanged	(0.32; 0.4)
0.8	1	0 · 1 = 0	unchanged	(0.32; 0.4)

Table 5.5: Trace of `best_bid(0.8, 2)`. *P(win) = min(1, b/0.5).*

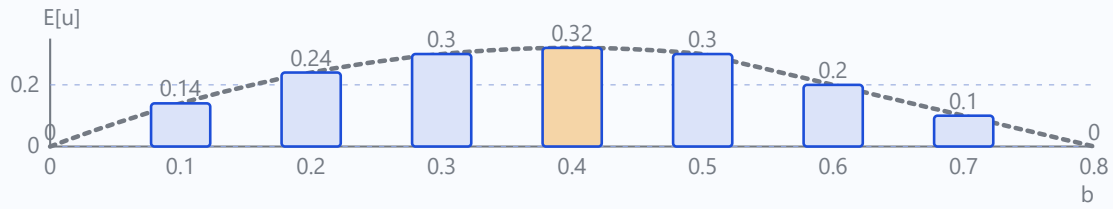


Figure 5.11: The end of the trace of `best_bid(0.8, 2)`. The bars are the expected utility for each *b*; the highest bar (orange) is at *b = 0.4*, with *E[u] = 0.32*.

The function returns `(0.4, 0.32)`: the best bid is $b = 0.4 = v/2$ and the expected utility 0.32. The result is the same as the derivation. Figure 5.11 shows the two ends of the trade-off: a low bid kills winning, a high bid kills the profit. After $b = 0.5$ the winning probability has reached 1, so only the profit shrinks and the bars shorten linearly.

5.3.7 Shading shrinks as competition grows

The equilibrium bid $b(v) = (n - 1)v/n$ depends on the number of bidders n. The amount of shading, that is, the difference between value and bid, is $v - b(v) = v/n$. As the number of bidders grows, the shading shrinks.

n	b(v)	v = 0.8 → b	shading v − b
2	1/2 · v	0.4	0.4
3	2/3 · v	0.53	0.27
5	4/5 · v	0.64	0.16
10	9/10 · v	0.72	0.08

Table 5.6: Equilibrium bid and shading by number of bidders (value *v = 0.8*).

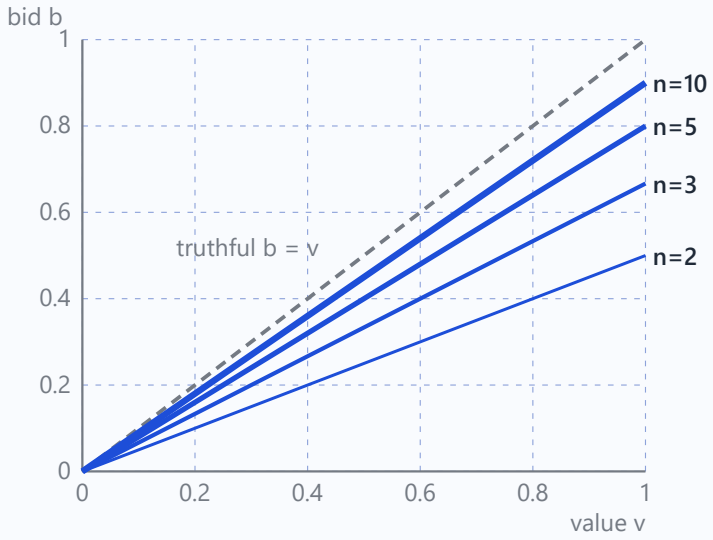


Figure 5.12: The equilibrium bid lines $b = (n-1)v/n$ for $n = 2, 3, 5, 10$. The dashed line is the truthful bid $b = v$. As n grows the lines approach the truthful bid.

Figure 5.12 shows the logic of the result. With many bidders, one of the rivals very probably has a value close to the bidder's; to win, the bid has to be close to the value. With few bidders the rival's value is very probably much lower and the bid can be shaded comfortably. In the limit $n \to \infty$, $b \to v$: in a crowded market the first-price rule too forces bidders to behave almost truthfully.

The interactive figure below shows the same relationship as the number of bidders changes. On the left are the truthful and equilibrium bid lines and the shading region between them; on the right is the expected revenue of the two rules. The default is $n = 2$.

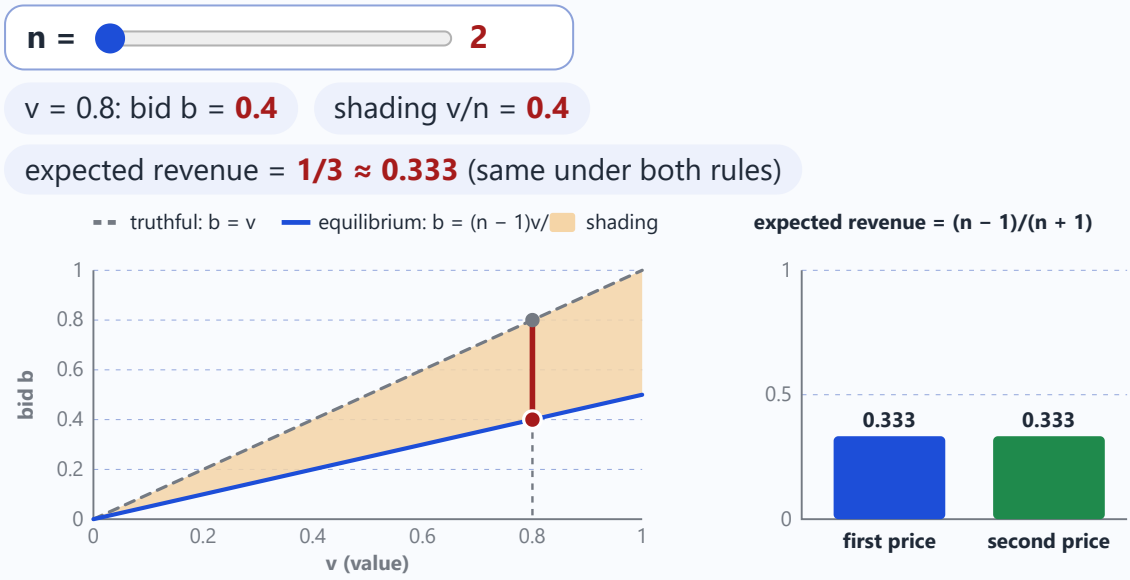


Figure 5.13: Interactive bid shading. The slider changes the number of bidders n from 2 to 10. On the left are the equilibrium bid and the shading v/n for $v = 0.8$; on the right is the expected revenue $(n-1)/(n+1)$ of the

first-price and second-price rules. At the default $n = 2$ the bid is 0.4, the shading is 0.4, and the expected revenue is 1/3 under both rules.

The second noteworthy fact in Figure 5.13 is the bars on the right: the expected revenue of the first-price and second-price rules is equal for every n and increases with n. It is 1/3 at $n = 2$, 1/2 at $n = 3$, and 9/11 at $n = 10$. The reason for this equality is revenue equivalence (5.4.2).

Example 5.8

There are $n = 4$ bidders and the values are $U[0, 1]$. What are the first-price equilibrium bid, the amount of shading, and the expected utility of the bidder whose value is 0.6?

1. Equilibrium bid: $b = (4 - 1)/4 \cdot 0.6 = 0.75 \cdot 0.6 = 0.45$.
2. Shading: $v - b = 0.6 - 0.45 = 0.15 = v/n$.
3. Winning probability: all three rivals' values must be below 0.6: $0.6^3 = 0.216$.
4. Expected utility: $0.15 \cdot 0.216 = 0.0324 = 0.6^4/4$.

5.4 Revenue and information

So far the question has been asked from the bidder's point of view: what should I bid? The seller's question is different: which rule earns me more? The answer is surprising: under standard assumptions, all the basic rules bring the same revenue on average. Which assumptions break this equality, how the seller can raise revenue (the reserve price), and the winner's curse that arises when information is common are all handled in the same framework.

5.4.1 Expected revenue with two bidders

We start with the simplest case: there are two bidders, and their values v_1, v_2 are independent $U[0, 1]$. Should the seller choose the second-price rule or the first-price rule? For each rule we first write down the seller's revenue in a single auction and then compute the expectation of that revenue.

Example 5.9

1. **Second price: revenue.** Everyone bids truthfully (Theorem 5.1). The winner is the bidder with the larger value and pays the smaller value. The seller's revenue is $\min(v_1, v_2)$.
2. **Second price: expectation.** If the smaller of the two values exceeds t, both exceed t. By independence, $P(\min > t) = (1 - t) \cdot (1 - t) = (1 - t)^2$. By the tail formula, $E[\min] = \int_0^1 (1 - t)^2 \, dt$. With the substitution $s = 1 - t$ the integral becomes $\int_0^1 s^2 \, ds = 1/3$.

3. **First price: revenue.** In equilibrium everyone bids $b = v/2$ (5.3.5). The winner is again the bidder with the larger value and pays their own bid, that is, half the larger value. The revenue is $max(v_1, v_2)/2$.
4. **First price: expectation.** If the larger of the two values is below t, both are: $P(max < t) = t^2$, so $P(max > t) = 1 - t^2$. By the tail formula, $E[max] = \int_0^1 (1 - t^2)\, dt = 1 - 1/3 = 2/3$. The revenue is half of this: $2/3 \cdot 1/2 = 1/3$.
5. **English and Dutch.** With private values the English auction gives the same result as second price: 1/3. The Dutch auction is strategically identical to first price: 1/3.
6. **Result.** The four rules give the same expected revenue: 1/3.

Rule	Seller's revenue	Expected revenue
Second price	$min(v_1, v_2)$	$E[min] = \int_0^1 (1 - t)^2\, dt = 1/3$
First price	$max(v_1, v_2)/2$	$E[max] = \int_0^1 (1 - t^2)\, dt = 2/3 \to 1/3$
English	≈ second price	1/3
Dutch	≡ first price	1/3
Simulation	200,000 draws	second price 0.333 · first price 0.333

Table 5.7: Two bidders, values independent $U[0, 1]$: expected revenue of the four rules.

The last row is the result of a Monte Carlo simulation: the computer draws two random values 200,000 times, computes the revenue of the two rules in each draw, and takes the average. Both averages are very close to 0.333. The simulation is a way of checking the result even for someone who does not know how to do the integral.

The equality holds only for *expected* revenue. In individual auctions the revenues differ. If the values are 0.9 and 0.2, second price brings 0.2 and first price $0.9/2 = 0.45$. If the values are 0.6 and 0.5, second price brings 0.5 and first price 0.3. When the values are close, second price earns more; when they are far apart, first price does. On average these differences exactly cancel.

5.4.2 The revenue equivalence theorem

The equality with two bidders is no coincidence; it is a special case of a more general theorem.

Theorem 5.2 (Revenue equivalence)

With independent private values, risk-neutral and symmetric bidders, and the same allocation rule (the highest value wins), standard auctions in which the lowest-value type pays zero yield the same expected revenue (Vickrey, 1961; Myerson, 1981; Riley and Samuelson, 1981).

Why the theorem is true is explained by Myerson's payment formula. In an auction, let the winning probability of a bidder with value v be $x(v)$ and the expected payment $p(v)$. If the bidders behave according to equilibrium, there is the following relation between the two:

$$p(v) = v \cdot x(v) - \int_0^v x(z)\, dz$$

On the right-hand side of the formula there is only the allocation rule $x(\cdot)$. The idea behind the formula is Myerson's lemma from mechanism design: since a bidder with value v could also choose to act as though their value were lower, in equilibrium the expected utility must rise with v exactly at the rate of the winning probability. This condition, once the utility of the lowest type (here 0) is fixed, determines the expected payment uniquely. The conclusion is: the same $x(\cdot)$ → the same $p(\cdot)$ → the same expected revenue. In all standard auctions the bidder with the highest value wins, so $x(\cdot)$ is the same; their revenues are therefore the same.

Example 5.10

Two bidders, $U[0, 1]$. The expected payment of the bidder with value v is computed in three ways.

1. **By the formula.** To win, the rival's value must be below v: $x(v) = v$. Then $p(v) = v \cdot v - \int_0^v z\, dz = v^2 - v^2/2 = v^2/2$.
2. **Under first price.** The bid is $v/2$ and the winning probability v; on winning the bid is paid. The expected payment is $v/2 \cdot v = v^2/2$.
3. **Under second price.** The winning probability is v; on winning the rival's value is paid. Given that the rival's value is below v, it is uniformly distributed on $[0, v]$ and its average is $v/2$. The expected payment is $v \cdot v/2 = v^2/2$.
4. **Passing to revenue.** The three routes give the same result. The seller's expected revenue is the sum of the two bidders' expected payments: $2 \cdot E[v^2/2] = E[v^2] = \int_0^1 t^2\, dt = 1/3$.

The same computation can be done for n bidders. Then $x(v) = v^{n-1}$, $p(v) = v^n - v^n/n = (n-1)v^n/n$, and the expected revenue is $n \cdot (n-1)/n \cdot E[v^n] = (n-1)/(n+1)$. The same result is reached from the second-price side: the expectation of the second-largest of n independent $U[0, 1]$ values is $(n-1)/(n+1)$. From the first-price side, the product of the expectation of the largest value, $n/(n+1)$, and the bid ratio $(n-1)/n$ is again $(n-1)/(n+1)$.

n	Theory	Second price (simulation)	First price (simulation)
2	1/3 = 0.33	0.332	0.333
3	1/2 = 0.5	0.500	0.500
4	3/5 = 0.6	0.600	0.600
5	2/3 = 0.67	0.668	0.667

Table 5.8: Expected revenue $(n-1)/(n+1)$ and simulation results.

Table 5.8 shows the two results together: for every n the two rules bring the same revenue and the revenue grows with n. The simulated values differ from the theory in the third decimal place; this is the natural random error of a finite number of draws.

The assumptions of the theorem matter. Risk aversion, budget constraints, common values, or asymmetric information can change the result. If bidders are not symmetric (for example, if one bidder's value is typically higher), then under first price the person with the highest value may not always win; the allocation rule changes and the equality breaks. The theorem therefore does not make the question "which rule earns more?" disappear; it moves the answer to the assumptions. In practice the auction format is chosen not by revenue alone but together with simplicity, transparency, and incentives to participate.

Common mistake

Reading revenue equivalence as "in every auction the two rules give the same price". The theorem is about *expected* revenue; in a single auction the revenues are often different (the example of 0.9 and 0.2 in 5.4.1). Another mistake is to generalize the theorem to all rules: a rule that gives the item to a random bidder, or does not give it to the highest-value bidder, makes a different allocation and brings a different revenue.

5.4.3 Changing the assumption: risk aversion

One assumption of revenue equivalence is that bidders are risk-neutral. What happens when this assumption is dropped? We build a model with two bidders and $U[0, 1]$ values. Let the winner's profit be $x = v - price$ and the utility be

$$u(x) = x^{\rho},\ 0 < \rho \le 1$$

Here $\rho = 1$ is the risk-neutral bidder: utility is the profit itself. $\rho < 1$ is a risk-averse bidder; for example for $\rho = ½$, $u(x) = \sqrt{x}$. The square-root curve is steep at small profits and flat at large ones. This means that a risk-averse person prefers a "small but certain" profit to a "large but uncertain" one: the first few lira of profit seem very valuable, the rest less so.

Second price does not change. The price is set by the rival's bid, not by one's own. The proof in 5.3.2 used nothing but the utility being an increasing function of profit; the truthful bid is therefore still weakly dominant. The revenue is again $E[min] = 1/3$.

Under first price the bid rises. If the rival bids $\beta\, v_r$, the probability of winning with bid b is b/β and the expected utility is $(v - b)^{\rho} \cdot b/\beta$. Taking the derivative gives $[-\rho\,(v - b)^{\rho-1}\, b + (v - b)^{\rho}]/\beta$. Setting it to zero and dropping the common factor $(v - b)^{\rho-1}$ gives $-\rho\, b + (v - b) = 0$, that is,

$$b(v) = v / (1 + \rho)$$

is obtained. For $\rho = 1$ it gives $v/2$, the familiar risk-neutral equilibrium; for $\rho = \frac{1}{2}$ it gives $2v/3$. The risk-averse bidder bids higher to reduce the risk of losing; in a sense, the bidder pays an "insurance premium" to raise the probability of winning.

Expected revenue	ρ = 1	ρ = ½
Second price	1/3 ≈ 0.333	1/3 ≈ 0.333
First price (E[max] = 2/3)	(2/3) · (1/2) = 1/3	(2/3) · (2/3) = 4/9 ≈ 0.444

Table 5.9: The effect of risk aversion on expected revenue (two bidders, $U[0, 1]$).

Table 5.9 summarizes the result: risk aversion breaks revenue equivalence, and the first-price rule earns the seller more. A seller who believes the bidders are risk-averse has a reason to choose the first-price rule (or the Dutch auction, which is identical to it).

Example 5.11

Two bidders, $\rho = \frac{1}{2}$. What does the bidder with value 0.6 bid under first price, and what is the utility if that bidder wins?

1. Equilibrium bid: $b = 0.6/(1 + 0.5) = 0.4$. A risk-neutral bidder would bid 0.3.
2. Winning probability: the rival bids $2v_r/3$; for $2v_r/3 < 0.4$ we need $v_r < 0.6$: the probability is 0.6.
3. On winning, the profit is $0.6 - 0.4 = 0.2$ and the utility $\sqrt{0.2} \approx 0.447$. The expected utility is $0.6 \cdot 0.447 \approx 0.268$.

5.4.4 Revenue maximization: Myerson and the reserve price

Revenue equivalence says that standard auctions bring the same revenue. But what should the seller do if the seller sets efficiency aside and wants only to maximize expected revenue? Roger Myerson answered this question (Myerson, 1981). The key to the answer is the concept of virtual value.

Definition 5.5

For a bidder whose value distribution is F with density f, the **virtual value** is

$$\psi(v) = v - (1 - F(v)) / f(v)$$

Here $F(v)$ is the probability that the value is below v, and $f(v)$ is the density. For $U[0, 1]$, since $F(v) = v$ and $f(v) = 1$, $\psi(v) = v - (1 - v) = 2v - 1$.

The virtual value measures how much revenue the seller can really extract from a bidder. The value v cannot be collected in full, because the bidder can hide the value and the seller has to leave the bidder an "information rent" as an incentive to be truthful. The subtracted term $(1 - F)/f$ is this rent. Myerson's basic result is this: *expected revenue equals the expectation of the winner's virtual value*. The mechanism that maximizes revenue therefore gives the item to the bidder with the largest virtual value; if even the largest virtual value is negative, it does not sell the item, because selling to someone with a negative virtual value reduces expected revenue.

If the virtual value is an increasing function and the bidders are symmetric, "the largest virtual value" and "the largest value" belong to the same person. The only difference is that no sale is made to those whose virtual value is negative. This is the **second-price auction with a reserve price**: the seller announces in advance a **reserve price** r; bids below r are invalid. The reserve is the point at which the virtual value is zero:

$$\psi(r) = 0 \Rightarrow 2r - 1 = 0 \Rightarrow r = 1/2$$

This threshold depends only on the value distribution; it is independent of the number of bidders n. For $U[0, 1]$ the best reserve is $1/2$ with two bidders and with ten.

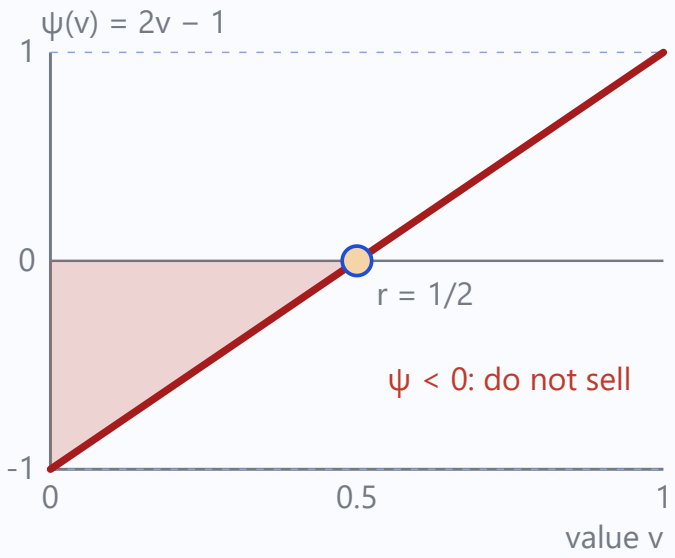


Figure 5.14: The virtual value $\psi(v) = 2v - 1$ for $U[0, 1]$. The line crosses zero at $v = 1/2$. In the pink region $\psi < 0$: selling to a bidder whose value is below $1/2$ reduces expected revenue; the seller does not sell.

The rules of the second-price auction with a reserve are as follows. If the highest bid is below r, the item is not sold. If only one bid is above r, the winner pays r. If two bids are above r, the winner pays the second-highest bid. In short, the winner pays $max(r, second\text{-}highest\ bid)$. The reserve is as if the seller had also placed a bid; truthful bidding is weakly dominant under this rule too.

With two bidders, the expected revenue of the second-price auction with a reserve can be computed with the virtual-value result. The revenue is the expectation of the winner's virtual value, and the item is sold only when the largest value reaches r (is at least r). The density of the larger of the two values is $2t$ (the derivative of $P(max < t) = t^2$). Then

$$E[revenue] = \int_r^1 (2t - 1) \cdot 2t \, dt = [4t^3/3 - t^2]_r^1 = 1/3 + r^2 - 4r^3/3$$

At $r = 0$ the revenue of the second price without a reserve comes out as 1/3. At $r = 1/2$ it is $1/3 + 1/4 - 1/6 = 5/12 \approx 0.417$. The reserve raises revenue by about 25%. This has a cost: sometimes there is no sale at all, and the item stays with the seller although there is a buyer who values it. This efficiency loss is computed in 5.4.6.

Example 5.12

Two bidders, $U[0, 1]$, reserve $r = 1/2$. The outcome is found for three pairs of values.

1. **(0.8, 0.3):** only 0.8 passes the reserve. The winner pays $max(0.5, 0.3) = 0.5$. Without a reserve the winner would have paid 0.3.
2. **(0.9, 0.7):** both pass the reserve. The winner pays $max(0.5, 0.7) = 0.7$; the reserve does not affect this auction.
3. **(0.4, 0.2):** both are below the reserve; the item is not sold. Without a reserve the bidder with 0.4 would have bought it for 0.2.

The reserve pushes the price up when the rival is weak (pair 1), but loses the sale when both bidders are weak (pair 3). On average the first effect dominates.

5.4.5 Code trace: best_reserve()

To find the best reserve numerically, two functions can be written. `revenue(r)` approximates on a grid the expected revenue of a two-bidder second-price auction with reserve r; `best_reserve()` tries the values $r = 0; 0.1; ...; 1$.

```
def revenue(r, m=100):      # 2 bidders, U[0,1]
  v = [(i + .5) / m for i in range(m)]
  total = sum(max(min(a, c), r) for a in v
              for c in v if max(a, c) >= r)
  return total / m**2       # mean payment
def best_reserve(step=0.1):
  best, r_best = -1, None
  for k in range(round(1/step) + 1):
    R = revenue(k * step)
    if R > best: best, r_best = R, k * step
  return r_best, best
```

The code works line by line as follows:

- `v = [(i + .5) / m for i in range(m)]`: the interval *[0, 1]* is divided into *m = 100* equal parts and the midpoint of each part is taken: 0.005; 0.015; …; 0.995. These points represent the uniform distribution.
- Two nested loops run over all value pairs *(a, c)*; there are *100 × 100 = 10,000* pairs and each is equally likely.
- `if max(a, c) >= r`: if the largest value does not reach the reserve there is no sale and the revenue is 0; these pairs do not take part in the sum.
- `max(min(a, c), r)`: if there is a sale, the winner pays the larger of the smaller value and the reserve.
- `total / m**2`: the total payment is divided by the number of pairs; pairs without a sale are counted with a contribution of 0. The result approximates the expected revenue.
- `best_reserve`: tries `r = k · step` for `k = 0, 1, …, 10`. `range(round(1/step) + 1)` ensures that *r = 1* is tried as well. When a larger revenue is found, the best is updated.

r	revenue(r)	Comparison	(best, r_best)	Formula
0	0.333	0.333 > −1: new best	(0.333; 0)	0.333
0.1	0.342	new best	(0.342; 0.1)	0.342
0.2	0.363	new best	(0.363; 0.2)	0.363
0.3	0.387	new best	(0.387; 0.3)	0.387
0.4	0.408	new best	(0.408; 0.4)	0.408
0.5	0.417	new best	(0.417; 0.5)	0.417
0.6	0.405	< 0.417: unchanged	(0.417; 0.5)	0.405
0.7	0.366	unchanged	(0.417; 0.5)	0.366
0.8	0.291	unchanged	(0.417; 0.5)	0.291
0.9	0.171	unchanged	(0.417; 0.5)	0.171
1	0	unchanged	(0.417; 0.5)	0

Table 5.10: Trace of `best_reserve()`. The last column is the formula $1/3 + r^2 - 4r^3/3$.

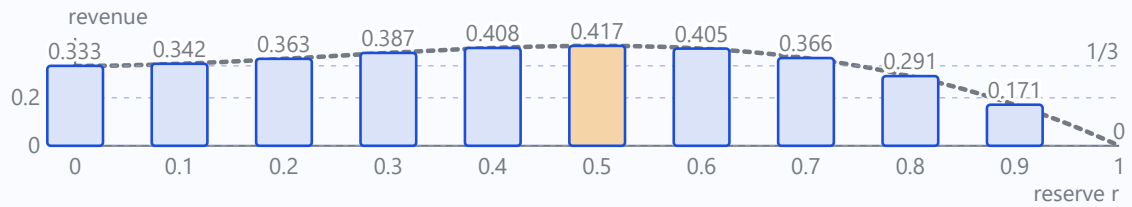


Figure 5.15: The end of the trace of `best_reserve()`. The bars are the expected revenue for each reserve; the highest bar (orange) is at *r = 0.5*, with revenue ≈ 0.417. The dashed curve is the formula.

The function returns $r = 0.5$ and revenue ≈ 0.417; this is the same as the theoretical result 5/12. Figure 5.15 shows that revenue first rises and then falls with the reserve. With no reserve ($r = 0$) the revenue is 1/3. High reserves often end with no sale: at $r = 1$ the item is never sold and the revenue is 0. The reason the grid computation agrees so well with the formula is that the midpoint rule gives a very small error for the uniform distribution.

5.4.6 The cost of a reserve

A well-chosen reserve can raise the seller's expected revenue (a very high reserve, for example $r = 1$, brings revenue down to zero), but it can reduce the total benefit society gets from the item, that is, **social welfare**. With two bidders and reserve r, three quantities are computed.

- **Expected revenue:** $1/3 + r^2 - 4r^3/3$ (5.4.4).
- **Probability of no sale:** both values must be below r: $r \cdot r = r^2$.
- **Expected loss of social welfare:** in the efficient outcome the bidder with the highest value gets the item and welfare equals this value. If this value is below r, the item is not sold and the value is lost. Since the density of the largest value is $2t$, the loss is $\int_0^r t \cdot 2t\, dt = 2r^3/3$.

r	Expected revenue	Probability of no sale	Expected welfare loss
0	1/3 ≈ 0.333	0	0
0.25	3/8 = 0.375	1/16 = %6.25	1/96 ≈ 0.010
0.5	5/12 ≈ 0.417	1/4 = %25	1/12 ≈ 0.083
0.75	1/3 ≈ 0.333	9/16 = %56.25	9/32 ≈ 0.281

Table 5.11: The effect of the reserve on revenue and efficiency (two bidders, $U[0, 1]$).

Social welfare with no reserve is $E[max] = 2/3$, the expectation of the largest value. According to Table 5.11, $r = 0.5$ maximizes revenue, but the item goes unsold with probability 25% and expected social welfare falls to $2/3 - 1/12 = 7/12$. The loss is $(1/12)/(2/3) = 1/8$, that is, 12.5%. The interests of the seller and of society diverge here: what is best for the seller is not best for society.

The last row of the table gives another lesson: raising the reserve too far also lowers revenue. At $r = 0.75$ revenue falls back to the no-reserve level (1/3), but the item goes unsold with more than half probability and the welfare loss is very large.

Common mistake

Thinking that "the higher the reserve, the higher the revenue". As the reserve rises, the winner's payment goes up, but the probability of a sale goes down. The two effects balance at $r = 1/2$; beyond it the second effect dominates. Another mistake is to think that the best reserve changes with the number of bidders: for $U[0, 1]$ the condition $\psi(r) = 0$ does not involve n.

5.4.7 All-pay and multi-unit auctions

In an **all-pay auction** the highest bid wins but *everyone* pays their bid. At first sight it looks odd, but it models many real contests: lobbying, R&D races, election campaigns. In all of these the prize is won by one person, but the effort and money spent do not return to the losers.

With two bidders and *U[0, 1]* values, in the symmetric equilibrium the bid is $b(v) = v^2/2$. This result follows easily from revenue equivalence. In the all-pay rule too the highest value wins, so the allocation rule is the same. The expected payment of a bidder with value v is therefore the same as under first and second price, $v^2/2$ (5.4.2). Since in all-pay the bid is paid in every case, the expected payment is the bid itself: $b(v) = v^2/2$. A bidder with value 0.8 bids 0.32; under first price the bid would be 0.4. Types with a low chance of winning hardly bid at all: a bidder with value 0.2 bids only 0.02.

The seller's expected revenue is again 1/3. This is a strong demonstration of revenue equivalence: however different the rules look, as long as the highest value wins, the average revenue does not change.

When values are equal and known to all, the all-pay model has an interesting result. For example, if two firms spend on lobbying for a prize of value V (e.g. a contract), in equilibrium the expected total spending equals the value of the prize. The entire value of the prize "dissipates" in the contest; this phenomenon is called rent dissipation and explains the cost of competitive lobbying in public policy.

Multi-unit auctions are auctions in which several units are sold at the same time. If the units are identical (for example government bonds), there are two basic rules. In a **uniform-price auction** every unit won is paid for at the same unit price; whoever takes two units pays twice what the one who takes one unit pays. In a **discriminatory (pay-as-bid) auction**, for each accepted unit the bid placed on that unit is paid. If the units are worth more together (for example frequency licenses for neighboring regions), bidders are allowed to place *package bids* for bundles of units: a bidder can bid a single price for a whole bundle. A package bid is a form of bidding; the payment rule is chosen separately. VCG is one of the payment rules that can be used in this setting; for example, core-selecting package auctions use the "core" prices that are closest to the VCG payments (Day and Milgrom, 2008). Choosing the winning bundles is also computationally hard.

There is another way to raise revenue. Bulow and Klemperer (1996) showed that, when values are independent and come from the same (well-behaved) distribution, attracting one extra bidder to an auction without a reserve earns as much as an auction run with the best reserve price, and often more. In the *U[0, 1]* example this is clearly visible: with two bidders and the best reserve the revenue is $5/12 \approx 0.417$; with three bidders and no reserve it is $(3 - 1)/(3 + 1) = 1/2$. For the seller, increasing competition can be worth more than fine-tuning the rules.

5.4.8 The winner's curse

In a common-value auction the item's true value is the same for everyone, but no one knows it. Each participant estimates the value and the estimates contain errors: some are above the true value, some below. The highest bid in the auction most probably comes from the participant who estimates the

value highest. By winning, this participant also learns that their own estimate is most probably too optimistic.

Definition 5.6

The winner's curse: in a common-value auction, because the winner bids without taking into account the information that winning carries (that their estimate is probably too high), the winner on average pays more than the item's true value and loses money.

The essence of the curse is a statistical fact: *even when estimates are correct on average, the largest estimate is systematically high*. The average of five estimates can equal the true value; but the largest of the five is almost always above the true value. The winner is exactly the owner of this largest estimate. To reduce this risk, participants keep their bids below their observations and take the quality of their information into account.

The phenomenon was first documented in oil lease auctions in the Gulf of Mexico. Engineers at the Atlantic Richfield company noticed that the fields they won systematically returned less than expected, and gave the phenomenon the name "winner's curse" (Capen, Clapp, and Campbell, 1971).

The size of the curse can be computed with a model. Let each firm's estimate be $V + e_i$ and let the estimation errors e_i be independent and $U[-20, 20]$. An error can be written as $-20 + 40\,u$, where $u \sim U[0, 1]$. Since the expectation of the largest of n numbers $U[0, 1]$ is $n/(n + 1)$, the expectation of the largest error is

$$-20 + 40 \cdot n/(n + 1) = 20(n - 1)/(n + 1)$$

The result is 6.7 with two firms, 13.3 with five, and 16.4 with ten. The curse grows as rivals increase, because the most optimistic member of a large group is more optimistic.

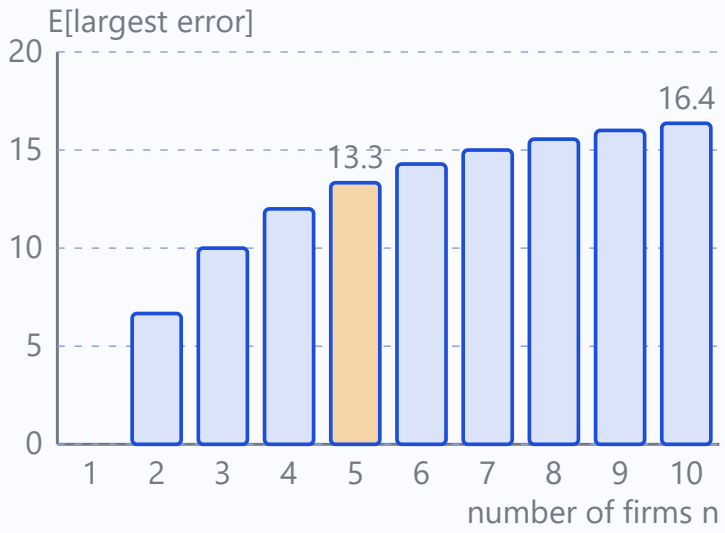


Figure 5.16: The winner's (largest estimate's) expected overestimate $20(n-1)/(n+1)$ by number of firms when the estimation errors are $U[-20, 20]$. It is 13.3 with five firms (orange) and 16.4 with ten. A simulation gives the same values.

Figure 5.16 yields a result that runs against intuition. With private values, the bid approaches the value as rivals increase (5.3.7). With common values, as rivals increase the bid *should move further from the estimate*, because winning means being the most optimistic of the crowd. The more rivals there are, the more one must shade.

5.4.9 An oil field auction, step by step

Example 5.13

1. **Setting.** The true value of the field is $V = 100$, but no one knows it. The five firms' estimates are 82, 91, 96, 107, 124. The average is $500/5 = 100$: the estimates are right on average.
2. **First price, unshaded bid.** If everyone bids their estimate, the most optimistic firm, F5, wins and pays 124. The profit is $100 - 124 = -24$.
3. **Second price, unshaded bid.** F5 wins again but pays the second-highest bid, 107. The profit is $100 - 107 = -7$. Even the second-price rule does not remove the curse: winning means saying yes to the question "am I the one who exaggerates the most?"
4. **Expectation.** The five estimates are a single realization; to see average behavior an error model is needed. If the errors are $U[-20, 20]$, the error of the largest of five estimates is on average $20 \cdot 4/6 \approx 13.3$. The expected profit of bidding one's estimate under first price is ≈ -13.3. The error of the second-largest estimate, which is the price paid under second price, is on average $20 \cdot (n-3)/(n+1) = 20 \cdot 2/6 \approx 6.7$; the expected profit is ≈ -6.7.

5. **Simulation.** In a first-price simulation of 100,000 auctions, the average profit of bidding one's estimate comes out as −13.3. When the bid is shaded by 13.3 below the estimate, the average profit is ≈ 0.
6. **Number of rivals.** With ten firms the required shading is *20 · 9/11 ≈ 16.4*: the more rivals, the more one must shade.

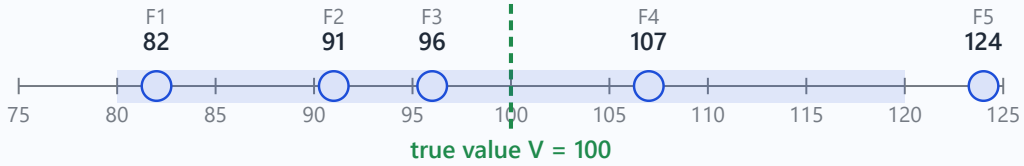


Figure 5.17: The oil field auction. The five firms' estimates (82, 91, 96, 107, 124) are scattered on both sides of the true value *V = 100* (green dashed line); the shaded band is the interval *[80, 120]*. The winner, F5, overestimated the true value by 24.

Case	Winner / payment	Profit
First price, unshaded bid	F5 → 124	−24
Second price, unshaded bid	F5 → 107	−7
Expectation of the largest error (n = 5)	20 · 4/6	≈ 13.3
Simulation, first price, unshaded bid	average	−13.3
Simulation, first price, shaded by 13.3	average	0

Table 5.12: Summary of the oil field example.

Figure 5.17 and Table 5.12 together show the logic of the curse. None of the firms is systematically optimistic; their estimates are scattered symmetrically on both sides of the true value. The problem is that the winner is not chosen at random: by definition, the winner is the one who made the largest estimate. A rational firm should ask, when placing a bid: "If I win with this bid, what does that say about my estimate?" Thinking conditionally on the event of winning removes the curse. The five estimates in the example are only an illustration: F5's error of +24 lies outside the *U[−20, 20]* model. If the error range were wider (e.g. *U[−30, 30]*), the expected overestimate would grow in the same proportion: with five firms, *30 · 4/6 = 20*.

Common mistake

Applying the winner's curse to private-value auctions. With private values the bidder knows their own value; what others think does not change it, and winning carries no bad news. The curse arises only when the value is common (or partly common) and estimates are noisy. Another mistake is to think that "the curse shrinks as rivals increase"; on the contrary, as rivals increase the largest estimate gets higher.

5.5 Two-sided markets

So far there has been a single seller and many buyers. In stock exchanges, however, both sides are crowded: many buyers and many sellers place orders at the same time. Such markets are organized by the double auction. The matching and price-setting rule, the incentive problems of this rule, and the McAfee rule as a solution to those problems are the basic pieces of this structure. Auctions are part of a wider field, market design.

5.5.1 The double auction

Definition 5.7

A **double auction** is a mechanism in which several buyers and several sellers place orders at the same time. Buyers submit **bids**: the highest amount they are willing to pay for one unit. Sellers submit **asks**: the lowest amount they will accept for one unit. The list of all orders is called the **order book**.

Stock exchanges, electricity markets, and carbon allowance markets have this structure. The mechanism does two jobs: it decides who trades with whom (matching) and it decides the price at which the trade takes place.

- **Matching:** bids are sorted from largest to smallest and asks from smallest to largest. The most eager buyer is matched with the cheapest seller, the second most eager buyer with the second cheapest seller, and so on. Pairs are matched as long as the bid is greater than or equal to the ask.
- **Price:** a single market price is chosen from the interval between the last matched pair and the first unmatched pair; for example, the midpoint of the interval. Everyone who is matched trades at this price.
- **Total surplus:** the value created by a trade is the difference between the buyer's value and the seller's cost. For the matched pairs, the sum $\Sigma\ (bid - ask)$ is called the **total surplus**. This surplus is the sum of the buyer surplus (buyer's value − price) and the seller surplus (price − seller's cost). The matching above gives the largest total surplus possible.

Why sorted matching gives the largest surplus can be seen as follows. If a bid is larger than an ask, that trade creates positive value. Matching the largest bids with the smallest asks means making the positive-value trades first. In the sorted lists the bids decrease and the asks increase, so the differences shrink pair by pair; stopping at the first negative difference means making all the positive-value trades and none of the negative-value ones.

5.5.2 Matching, step by step

The order book has five buyers and five sellers. The buyers are willing to pay 70, 40, 90, 55, and 80; the sellers are willing to accept 60, 85, 35, 75, and 50.

Example 5.14

1. **Sorting.** The bids in decreasing order: 90, 80, 70, 55, 40 (the demand curve). The asks in increasing order: 35, 50, 60, 75, 85 (the supply curve).
2. **Pair 1:** 90 ≥ 35 → matched; the surplus this trade creates is *90 − 35 = 55*.
3. **Pair 2:** 80 ≥ 50 → matched; surplus *80 − 50 = 30*.
4. **Pair 3:** 70 ≥ 60 → matched; surplus *70 − 60 = 10*.
5. **Pair 4:** 55 < 75 → the buyer does not pay what the seller asks; stop. 3 trades take place.
6. **Price interval.** The lower bound is the larger of the last matched seller's ask and the first unmatched buyer's bid: *max(60, 55) = 60*. The upper bound is the smaller of the last matched buyer's bid and the first unmatched seller's ask: *min(70, 75) = 70*. The interval is *[60, 70]*, and the midpoint price is 65.
7. **Surpluses.** The buyer surplus is *(90 − 65) + (80 − 65) + (70 − 65) = 45*. The seller surplus is *(65 − 35) + (65 − 50) + (65 − 60) = 50*. The total is *95 = 55 + 30 + 10*, the largest possible total surplus.

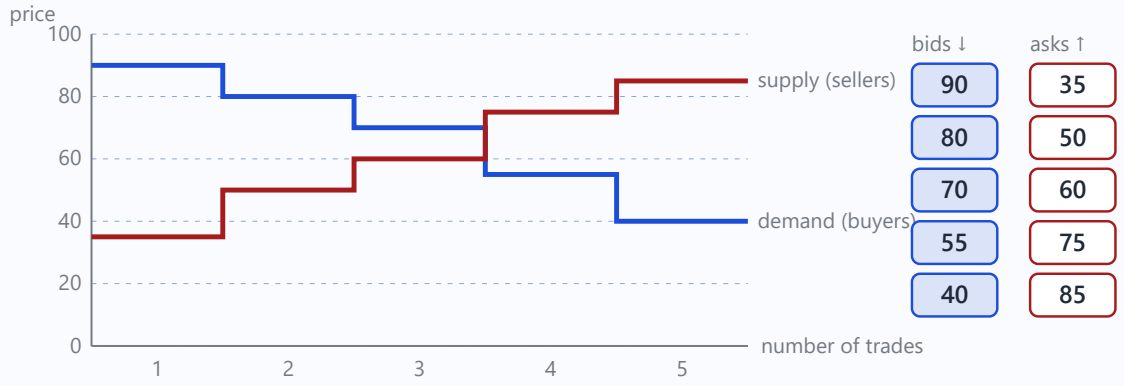


Figure 5.18: The sorted order book. The blue step line is the decreasing bids (the demand curve), the red step line the increasing asks (the supply curve). The sorted lists are shown on the right.

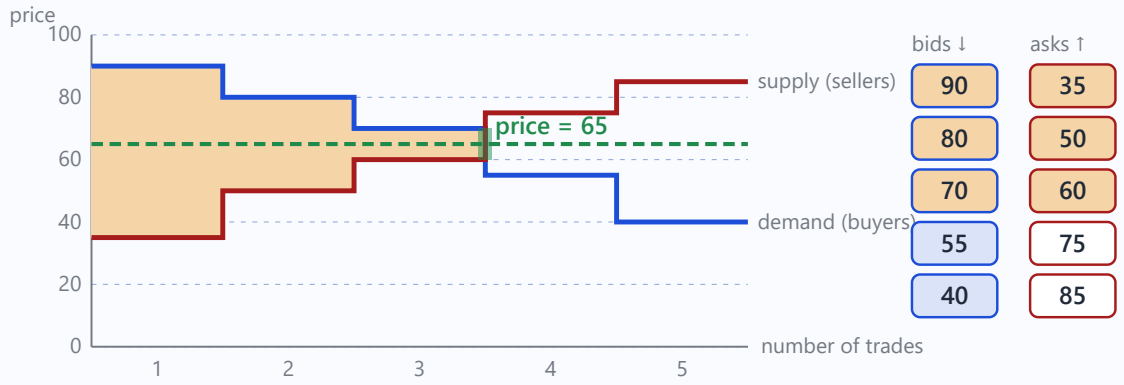


Figure 5.19: The end of the matching. The orange area is the first three trades, where the demand curve stays above the supply curve, and shows the total surplus (95). The green dashed line is the market price 65; the orange part above the line is the buyer surplus, the part below it the seller surplus.

Figure 5.18 and Figure 5.19 show the matching with the supply and demand curves familiar from economics. The demand curve (bids) is a decreasing step line and the supply curve (asks) an increasing one. Where the two curves cross determines the number of trades and the price interval. At every step before the crossing, demand is above supply: the trade creates value. After the crossing, supply rises above demand: the trade destroys value.

The logic of the two ends of the price interval is important. The lower bound is 60, because if the price falls below 60 the last matched seller (cost 60) loses money and does not want to sell. Also, if the price fell below 55, the buyer left out (55) would want to buy as well and the market would not be in equilibrium. The upper bound is 70, because if the price rises above 70 the last matched buyer does not want to buy; if it rises above 75 the seller left out would want to sell as well. Any price in the interval keeps these four people in place.

5.5.3 Code trace: double_auction()

The matching and the midpoint price can be written as a short function.

```
def double_auction(bids, asks):
  b = sorted(bids, reverse=True)  # buyers
  s = sorted(asks)                # sellers
  k, n = 0, min(len(b), len(s))
  while k < n and b[k] >= s[k]:
    k += 1                    # pair matched
  if k == 0:
    return 0, None
  lo = max([s[k-1]] + b[k:k+1])
  hi = min([b[k-1]] + s[k:k+1])
  return k, (lo + hi) / 2
```

The code works line by line as follows:

- `b = sorted(bids, reverse=True)`: the bids are sorted from largest to smallest; the most eager buyer comes first.
- `s = sorted(asks)`: the asks are sorted from smallest to largest; the cheapest seller comes first.
- `k` is the number of matched pairs; `n` is the length of the shorter list. The numbers of buyers and sellers can differ; the excess of the longer list cannot be matched anyway.
- `while k < n and b[k] >= s[k]`: if the next buyer meets the next seller's ask, the pair is matched and `k` increases by one.
- `if k == 0`: if there are no trades there is no price either; `(0, None)` is returned.
- `lo = max([s[k-1]] + b[k:k+1])`: the larger of the last matched seller's ask and (if there is one) the first unmatched buyer's bid. `b[k:k+1]` is a slice: it gives a one-element list if `k` is inside the list and an empty list if outside. The code therefore does not fail when all buyers are matched.
- `hi = min([b[k-1]] + s[k:k+1])`: the smaller of the last matched buyer's bid and (if there is one) the first unmatched seller's ask.
- The function returns the number of trades and the midpoint of the interval.

Step	Operation	Outcome
1	`b = sorted(…, reverse=True)`	[90, 80, 70, 55, 40]
2	`s = sorted(…)`	[35, 50, 60, 75, 85]
3	`k, n = 0, 5`	k = 0
4	b[0] = 90 ≥ s[0] = 35?	yes → k = 1
5	b[1] = 80 ≥ s[1] = 50?	yes → k = 2
6	b[2] = 70 ≥ s[2] = 60?	yes → k = 3
7	b[3] = 55 ≥ s[3] = 75?	no → loop ends
8	`k == 0` ?	no; the price is computed
9	`lo = max(s[2], b[3])`	max(60, 55) = 60
10	`hi = min(b[2], s[3])`	min(70, 75) = 70
11	return	(3, 65.0)

Table 5.13: Trace of `double_auction([70, 40, 90, 55, 80], [60, 85, 35, 75, 50])`.

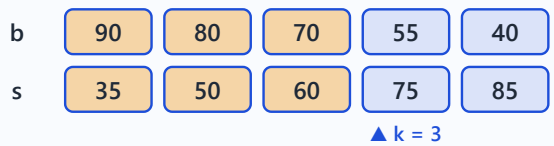


Figure 5.20: The end of the trace of `double_auction()`. The top row is the sorted bids `b`, the bottom row the sorted asks `s`; the orange cells are the three matched pairs and the loop stops at `k = 3`.

Figure 5.20 shows the result as two arrays. The function returns 3 trades and the price 65; this is the same as the hand computation. The cost of the algorithm is set by the sorting: $O(m \log m)$ for m orders. The matching loop passes over the lists once and is therefore linear.

5.5.4 Incentives in a double auction

In a one-sided auction the Vickrey rule was both efficient and strategy-proof. Is it possible to obtain the same in a two-sided market? The four properties we want are:

- **Efficiency:** all value-creating trades are made.
- **Incentive compatibility** (IC): for everyone, placing an honest order is best.
- **Individual rationality** (IR): no one loses by taking part.
- **Budget balance**: the market maker puts in no money from outside; what the buyers pay goes to the sellers.

> **Theorem 5.3 (Myerson–Satterthwaite, 1983)**
>
> In bilateral trade where the buyer's value and the seller's cost are independent and come from overlapping ranges, efficiency, (Bayesian) incentive compatibility, individual rationality, and budget balance cannot all hold together.

This is an impossibility result: at least one of the four properties must be given up. Practical rules differ in which one they give up.

The **midpoint rule** is efficient and budget balanced but not incentive compatible. The last matched buyer or seller can shift the price in their own favor by changing their order. In the example the last matched seller's cost is 60, and with an honest order the seller sells at 65 and gets a surplus of 5. What if the seller asks 64 instead of 60? Since $70 \geq 64$ the trade still takes place, but the interval is $[\max(64, 55), \min(70, 75)] = [64, 70]$ and the price becomes 67. The seller's surplus is $67 - 60 = 7 > 5$: lying pays off. The last matched buyer can do the same by lowering the bid. Exaggeration has a limit: if the seller asks above 70, the trade is lost. This is the same as the shading trade-off under first price.

The **McAfee double auction** sometimes gives up one trade to secure incentive compatibility (McAfee, 1992). The rule is as follows. The matching is done as above and k efficient trades are found. The average p_0 of the orders of the first unmatched buyer and the first unmatched seller is computed.

- If p_0 falls between the last matched seller's ask and the last matched buyer's bid, all k trades take place at p_0.
- If it does not, the last pair (the one bringing the least surplus) is removed. In the remaining $k - 1$ pairs the buyers pay the removed buyer's bid and the sellers receive the removed seller's ask.

The intuition for why the rule is incentive compatible is that no one who trades can set their own price. (The full proof also examines how the changing of one's order changes whether one trades at all; McAfee, 1992.) In the first case the price is set by the pair left out. In the second case the prices are set by the removed pair, and that pair does not trade. This is the two-sided version of the Vickrey idea that "the rival sets your payment". Honesty is dominant; the price is that in the second case one trade is lost. In the second case the buyers pay more than the sellers receive; the difference stays with the market maker. The rule thus satisfies budget balance in the weak sense: the market maker puts in no money from outside, but sometimes has money left over.

In our example the first pair left out is 55 (buyer) and 75 (seller). $p_0 = (55 + 75)/2 = 65$. This value lies in the interval $[60, 70]$; all three trades take place at 65 and no trade is lost. In this example the McAfee rule gives the same result as the midpoint rule.

5.5.5 Code trace: mcafee()

The McAfee rule can be written as follows:

```
def mcafee(bids, asks):
  b = sorted(bids, reverse=True) + [float('-inf')]
  s = sorted(asks) + [float('inf')]
  k = 0
  while b[k] >= s[k]:
    k += 1                # efficient trade
  if k == 0: return 0, None, None
  p0 = (b[k] + s[k]) / 2   # excluded pair
  if s[k-1] <= p0 <= b[k-1]:
    return k, p0, p0      # one price
  return k - 1, b[k-1], s[k-1]  # two prices
```

The code has two details:

- **Sentinels.** Minus infinity (`float('-inf')`) is appended to the end of the bids list and infinity (`float('inf')`) to the end of the asks list. A buyer bidding minus infinity cannot meet any ask, not even that of a seller who asks 0; no buyer can meet a seller who asks infinity. The loop thus stops by itself at the end of the shorter list, and `b[k]` and `s[k]` are always defined. (Had the buyer sentinel been 0, it would match a seller asking 0 and the loop would run off the end of the list.) If no real pair is left out, `p0` comes out infinite or undefined (`nan`), the price check fails, and the weakest pair is removed. This takes the place of the slice trick in `double_auction()` .
- **Two return forms.** The function returns `(number of trades, buyer price, seller price)` . In the single-price case the two prices are the same; in the second case they differ.

Consider a new example: buyers 90, 80, 70, 40; sellers 30, 50, 65, 85.

Step	Operation	Outcome
1	sorting and sentinels	b = [90, 80, 70, 40, −∞], s = [30, 50, 65, 85, ∞]

Step	Operation	Outcome
2	b[0] = 90 ≥ s[0] = 30?	yes: efficient pair, k = 1
3	b[1] = 80 ≥ s[1] = 50?	yes: k = 2
4	b[2] = 70 ≥ s[2] = 65?	yes: k = 3
5	b[3] = 40 ≥ s[3] = 85?	no: loop ends
6	`p0 = (b[3] + s[3]) / 2`	(40 + 85)/2 = 62.5
7	s[2] = 65 ≤ 62.5 ≤ b[2] = 70?	no: 62.5 < 65
8	last pair (70, 65) is removed	(2, 70, 65)

Table 5.14: Trace of `mcafee([90, 80, 70, 40], [30, 50, 65, 85])`.

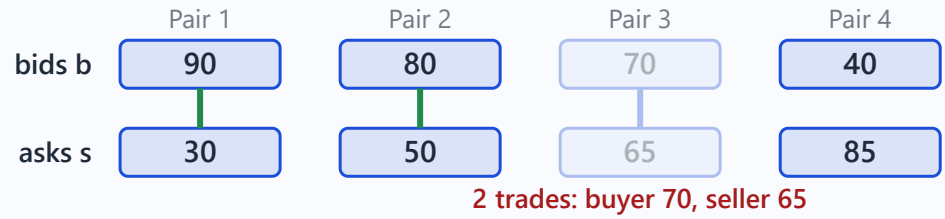


Figure 5.21: The end of the trace of `mcafee()`. The third pair (70, 65) is shown faded: it has been removed. In the remaining two trades the buyers pay 70 and the sellers receive 65.

The meaning of the check in step 7 is this: if trades were made at the price $p_0 = 62.5$, the third seller (cost 65) would lose money. The weakest pair (70, 65) is therefore removed. In the remaining two trades the buyers pay 70 and the sellers receive 65; the $70 - 65 = 5$ left over in each trade, $2 \cdot 5 = 10$ in all, stays with the market maker (Figure 5.21).

Had the same orders been given to the midpoint rule, `double_auction()` would have returned three trades and the price $(65 + 70)/2 = 67.5$. The total surplus would have been $60 + 30 + 5 = 95$. Under the McAfee rule the total surplus is $60 + 30 = 90$; the 5 units of surplus go with the lost third trade. This is the price of honesty, and it is at most one trade. In markets with many participants, the loss of one trade is a small fraction of the total surplus, so the rule is attractive in practice.

5.5.6 Auction and market design

Market design is the work of building the rules that determine with whom and under what conditions participants are matched. The basic goals are efficiency, incentive compatibility, fairness, stability, and feasibility. The rules should be strategy-proof: it should not pay participants to lie in order to turn the rules to their advantage.

Auctions are market design tools that allocate scarce resources according to prices and bids. Electricity markets, emission permits, public tenders, and spectrum licenses are applications with prices. In markets without money (for example, placing students in schools or kidney exchange for organ transplants), priority and matching rules take the place of price. In both cases the designer

faces the same trade-offs: revenue, efficiency, simplicity, and strategy-proofness cannot always be obtained together. The examples in the chapter show this clearly: the reserve gives up efficiency for revenue, and the McAfee rule gives up one trade for honesty.

5.5.7 From theory to practice: a short history

An old institution. Auctions are very old. Herodotus describes auctions held in Babylon in the 5th century BC. In AD 193 in Rome, the Praetorian Guard put the imperial throne up for auction and sold it to Didius Julianus; Julianus was killed two months later. This is regarded as one of the most striking examples of the winner's curse in history.

Theory. Auction theory began in 1961 with Vickrey's paper. Vickrey compared the four formats and found the first form of revenue equivalence (Vickrey, 1961). Myerson defined the revenue-maximizing auction (Myerson, 1981). Both received the Nobel Prize in Economics: Vickrey in 1996, Myerson in 2007.

Spectrum auctions. In 1994 the US Federal Communications Commission (FCC) held the first spectrum auction using the *simultaneous multiple-round auction* proposed by Paul Milgrom, Robert Wilson, and Preston McAfee. In this design many licenses are sold at the same time and in rounds; in each round bidders update their bids by looking at the prices of the other licenses. Milgrom and Wilson received the Nobel Prize in Economics in 2020.

Matching markets. Alvin Roth and Lloyd Shapley received the Nobel Prize in Economics in 2012 for their work on stable matching and swap rules in markets where price is not used.

A designer's tools are not limited to the allocation and payment rules. Information disclosure is also a tool: which bids are shown, and when? Participation conditions (who may bid, whether a deposit is required) are likewise tools. Real designs are tested with laboratory experiments and field data before being put into practice.

5.6 Review

In practice the chapter's results are weighed by two questions: why is the Vickrey auction, the most elegant result of the theory, rarely used, and how fair is each auction? A summary of the four formats and practice problems complete this discussion.

5.6.1 Why are Vickrey auctions rarely used?

The theoretical properties of the Vickrey auction are very attractive: truthful bidding is dominant, the item goes to the highest-value bidder, and revenue is the same as under the standard rules. Even so, the pure second-price sealed-bid auction is rarely seen in practice. Rothkopf, Teisberg, and Kahn (1990) and Ausubel and Milgrom (2006) discuss the reasons.

- **Trust.** Participants must trust that the second-highest bid is reported truthfully. Only the auctioneer sees the sealed envelopes; the seller can add a fake "second bid" and pull the price down to just under the winner's bid.
- **Intuitiveness.** The rule "pay the second price" does not feel natural to most people. In the lab, overbidding is common (5.3.3).
- **Model assumptions.** With common or interdependent values, the result that the truthful bid is dominant can lose its validity.
- **Many items.** Budget constraints and complementary goods complicate the payment rule. In VCG, the multi-item generalization, revenue can be very low and bidders can obtain illegitimate gains with false identities.
- **The seller's concern.** The difference between the winner's bid and the price paid can be large. A winner who bids 1 million and pays 100 thousand may be perceived in public as a "sale at a low price". There is also the risk of collusive bidding (secret agreement) among bidders.

For these reasons practical designs often strike a balance between the incentive for honesty and ease of implementation. The English auction gives the same result with private values in a transparent way: the price forms in plain view and no one has to trust an undisclosed bid. The English auction therefore works in practice as the "visible" version of the Vickrey auction.

5.6.2 Discussion: which auction, and how fair?

From a participant's point of view the formats offer different experiences. The English auction is transparent; seeing the price step by step is an advantage. In the Vickrey auction the truthful bid is best and no strategizing is needed. The Dutch and first-price auctions are fast, but the amount of shading must be guessed. Which format is preferred depends on personal strategy and appetite for risk: a risk-averse person shades less under first price, as seen in 5.4.3, and raises the probability of winning.

The question of fairness is more complex. On the one hand, setting the rules in advance provides predictability and can reduce manipulation; everyone competes under the same rules. On the other hand, players with large budgets have an advantage: an auction measures ability to pay, not need. Access to participation and inequality of information also matter: a bidder with better information avoids the curse in a common-value auction while less informed rivals lose. The seller's revenue and equality of opportunity are different goals. As the reserve price example showed, what is good for one goal may be bad for another. No rule guarantees "full fairness for everyone".

5.6.3 The four formats side by side, and practice problems

Format	Pays	Equilibrium bid	Equivalent
English	the price at which the last rival dropped out	stay in up to your value	second price
Second price	second-highest bid	$b = v$ (weakly dominant)	English
Dutch	the accepted price	$b = (n-1)v/n$	first price

Format	Pays	Equilibrium bid	Equivalent
First price	own bid	$b = (n-1)v/n$	Dutch

Table 5.15: The four formats side by side (IPV, $U[0, 1]$, n bidders).

Three additional results accompany Table 5.15. In all four formats the expected revenue is $(n - 1)/(n + 1)$; 1/3 with two bidders. With common values the winner may overpay; the more rivals there are, the more one must shade. In a two-sided market, bids are sorted in decreasing order and asks in increasing order and matched, and the price is chosen from the interval $[lo, hi]$.

The following practice problems review the chapter's basic computations.

Example 5.15

1. **The four rules.** The limits are 40, 65, 80, 90 and everyone bids their limit. The bidder with 90 wins. English ≈ 80, second price 80; Dutch 90, first price 90 (5.2.5).
2. **Equilibrium bid.** $n = 4$, value 0.6: $b = (4 - 1)/4 \cdot 0.6 = 0.45$.
3. **Expected revenue.** Three bidders, second price: $(n - 1)/(n + 1) = 2/4 = 1/2$.
4. **Overbidding in Vickrey.** Value 50, the rivals' highest bid 55, bid 60: the bidder wins but pays 55; utility $50 - 55 = -5$. With the truthful bid the bidder loses and the utility is 0.
5. **Two-sided.** Buyers 100, 70, 45; sellers 30, 60, 80. The pairs 100 ≥ 30 and 70 ≥ 60 are matched; we stop at 45 < 80: 2 trades. The interval is $[\max(60, 45), \min(70, 80)] = [60, 70]$, the price 65.
6. **Reserve.** Two bidders, $U[0, 1]$, second price, $r = 0.4$: $1/3 + 0.16 - 4 \cdot 0.064/3 = 1/3 + 0.16 - 0.0853 \approx 0.408$. This is more than 1/3 ≈ 0.333 with no reserve. The best is $r = 0.5$: 5/12 ≈ 0.417.

5.7 Summary

- An **auction** is a combination of an allocation rule (who gets it?) and a payment rule (what do they pay?). A bidder's **value** is private information and the **bid** is a strategic decision; utility is $v - p$ on winning and 0 on losing. The basic model is **independent private values** (IPV); the common-value model leads to the winner's curse.
- **The four basic formats:** English (the price rises; the last one remaining pays the price at which the last rival dropped out), Dutch (the price falls; the first to accept pays that price), first-price sealed-bid (pays their own bid), second-price/Vickrey (pays the second-highest bid). With private values English ≈ second price; in every case Dutch ≡ first price.
- **Vickrey's theorem:** in a second-price auction, the truthful bid is a weakly dominant strategy. The bid determines only whether you win, and the rivals determine the payment; for every B the truthful bid gives $\max(v - B, 0)$.

- **Bid shading:** under first price, the trade-off $E[u(b)] = (v - b) \cdot P(\text{win})$ gives the bid $b = (n-1)v/n$ in the symmetric Bayes–Nash equilibrium; $v/2$ with two bidders. The shading is v/n and shrinks as competition grows.
- **Revenue equivalence:** under IPV, risk-neutral symmetric bidders, and the same allocation rule, standard auctions have the same expected revenue: $(n-1)/(n+1)$, 1/3 with two bidders. The reason is Myerson's payment formula: $p(v) = v\,x(v) - \int_0^v x(z)\,dz$. The all-pay auction ($b = v^2/2$) brings the same revenue as well.
- **Risk aversion** breaks the equality: under first price the bid rises to $v/(1+\rho)$ and the revenue rises to 4/9 at $\rho = ½$; second price stays at 1/3.
- **Myerson's reserve price:** expected revenue is the expectation of the winner's virtual value; for $U[0, 1]$, $\psi(v) = 2v - 1$ and the best reserve is $r = 1/2$ (independent of n). With two bidders the revenue is $1/3 + r^2 - 4r^3/3$, 5/12 at the best reserve. The cost: with probability 25% there is no sale, and social welfare falls from 2/3 to 7/12.
- **The winner's curse:** with common values the winner is the one with the most optimistic estimate; if errors are $U[-a, a]$, the expected overestimate is $a(n-1)/(n+1)$. The more rivals, the more one must shade.
- **The double auction:** bids in decreasing order and asks in increasing order are matched; the price is chosen from the interval $[lo, hi]$ and the total surplus is the largest. By Myerson–Satterthwaite, with values and costs from independent, overlapping ranges, efficiency, IC, IR, and budget balance cannot all hold together; the midpoint rule is not IC, and the McAfee rule makes honesty dominant at the cost of one trade.

Concept	What it says	Key number / formula
Vickrey auction	Truthful bid is weakly dominant	$b = v$
Bid shading	Bid below the value under first price	$b = (n-1)v/n$
Revenue equivalence	Same allocation → same expected revenue	$(n-1)/(n+1)$
Risk aversion	First price earns more	$b = v/(1+\rho)$
Virtual value and reserve	Do not sell at a negative virtual value	$\psi(v) = 2v - 1$, $r = 1/2$, 5/12
The winner's curse	With common values the winner overpays	$20(n-1)/(n+1)$
Double auction	Sorted matching, a single price	midpoint of $[lo, hi]$
McAfee rule	Sometimes a lost trade for honesty	p_0 = average of the pair left out

Table 5.16: The chapter's concepts at a glance.

5.8 Key terms

auction bidder value bid valuation quasi-linear utility allocation rule payment rule price discovery independent private values (IPV) common value interdependent values English (ascending) auction Dutch (descending) auction sealed bid first-price sealed-bid auction

second-price (Vickrey) auction weakly dominant strategy truthful bid strategy-proof bid shading Bayes–Nash equilibrium revenue revenue equivalence theorem revenue maximization optimal auction (Myerson) virtual value reserve price all-pay auction multi-unit auction uniform-price auction discriminatory (pay-as-bid) auction spectrum auction winner's curse double auction bid (buy order) ask (sell order) order book surplus buyer / seller surplus midpoint rule McAfee double auction market maker budget balance market design Vickrey–Clarke–Groves (VCG) mechanism

---

5.9 Test Yourself

1. Five bidders have limits 35, 62, 78, 81, and 50; everyone bids their limit (in the Dutch auction the price falls continuously). Under the four rules, who wins, what do they pay, and what is the utility?

   Answer

   The winner under every rule is the bidder with limit 81. English: the last rival drops out at 78, payment ≈ 78, utility ≈ 3. Second price: the payment is the second-highest bid, 78; utility 3. Dutch: accepts when the price falls to 81, payment 81, utility 0. First price: payment 81, utility 0. English and second price give the same result, as do Dutch and first price.

2. In a Vickrey auction a bidder's value is $v = 40$. The rivals' highest bid B may be 20, 35, 50, or 60. What is the utility in each case for the truthful bid, for $b = 30$, and for $b = 55$?

   Answer

   Truthful (40): $B = 20 \to$ wins, pays 20, utility 20; $35 \to$ utility 5; 50 and 60 $\to$ loses, 0. $b = 30$: $20 \to 20$ (same); $35 \to$ loses, 0 (5 worse than truthful); 50, 60 $\to$ 0 (same). $b = 55$: $20 \to 20$; $35 \to 5$ (same); $50 \to$ wins, pays 50, utility −10 (worse than truthful); $60 \to 0$ (same). No deviation is better than the truthful bid at any B; the cases where it is worse are those where B falls between b and v.

3. In a first-price auction there are three bidders; two rivals bid 2/3 of their values and values are $U[0, 1]$. The third bidder's value is 0.9. What is the expected utility of the bids $b = 0.3; 0.45; 0.6; 0.75$? Which is best?

   Answer

   The probability of beating one rival is $\min(1, b/(2/3)) = \min(1, 1.5b)$; for two rivals it is squared. $b = 0.3$: $P = 0.45^2 = 0.2025$, $E[u] = 0.6 \cdot 0.2025 \approx 0.122$. $b = 0.45$: $P = 0.675^2 \approx 0.456$, $E[u] \approx 0.45 \cdot 0.456 \approx 0.205$. $b = 0.6$: $P = 0.9^2 = 0.81$, $E[u] = 0.3 \cdot 0.81 = 0.243$. $b = 0.75$: $P = 1$, $E[u] = 0.15$. The best is $b = 0.6 = (3 - 1)/3 \cdot 0.9$, the same as the equilibrium bid.

4. Three bidders, values $U[0, 1]$. By Myerson's payment formula, what is the expected payment of the bidder with value v? In the all-pay auction, what is the equilibrium bid of the bidder with value 0.9?

Answer

To win, both rivals' values must be below v: $x(v) = v^2$. $p(v) = v \cdot v^2 - \int_0^v z^2\,dz = v^3 - v^3/3 = 2v^3/3$. Check: under first price the bid is $2v/3$, the winning probability v^2, and the product $2v^3/3$. In all-pay the bid is paid in every case, so the bid equals the expected payment: $b(0.9) = 2 \cdot 0.729/3 = 0.486$.

5. Two bidders, values $U[0, 1]$, utility $u(x) = x^\rho$ with $\rho = 1/3$. What are the equilibrium bid under first price and the seller's expected revenue? How large is the difference from second price?

Answer

$b(v) = v/(1 + \rho) = v/(4/3) = 3v/4$. The winner is the bidder with the larger value; the revenue is $3/4 \cdot E[\max] = 3/4 \cdot 2/3 = 1/2$. Under second price the truthful bid remains dominant and the revenue is 1/3. First price earns $1/2 - 1/3 = 1/6$ more; the difference grows as risk aversion increases (as ρ gets smaller).

6. Two bidders, $U[0, 1]$, a second-price auction with reserve $r = 0.3$. What are the expected revenue, the probability of no sale, and the expected loss of social welfare? Would the best reserve change with five bidders?

Answer

The revenue is $1/3 + 0.09 - 4 \cdot 0.027/3 = 1/3 + 0.09 - 0.036 \approx 0.387$. The probability of no sale is $0.3^2 = 0.09$. The welfare loss is $2 \cdot 0.027/3 = 0.018$. The best reserve comes from $\psi(r) = 2r - 1 = 0$ and does not involve the number of bidders; with five bidders too it is $r = 1/2$.

7. In a common-value auction the estimation errors are $U[-15, 15]$. If three firms bid their estimates, what is the winner's expected profit under first price and under second price? What is the answer for first price with six firms?

Answer

The expectation of the largest error is $15(n - 1)/(n + 1)$; for $n = 3$ it is $15 \cdot 2/4 = 7.5$. Under first price the winner pays their own estimate: expected profit ≈ −7.5. Under second price the second-largest estimate is paid; the expectation of the second-largest error is $15(n - 3)/(n + 1) = 0$: expected profit 0. With six firms the first-price expected profit is $-15 \cdot 5/7 \approx -10.7$; the curse grows as rivals increase.

8. Buyers 75, 62, 58, 30; sellers 20, 41, 55, 70. How many trades and which prices do the midpoint

rule and the McAfee rule give? How much surplus is lost under the McAfee rule?

Answer

The sorted lists are already decreasing/increasing: 75 ≥ 20, 62 ≥ 41, 58 ≥ 55 are matched; we stop at 30 < 70, $k = 3$. Midpoint: the interval is $[\max(55, 30), \min(58, 70)] = [55, 58]$, price 56.5; three trades. McAfee: $p_0 = (30 + 70)/2 = 50$; 50 is not in $[55, 58]$. The last pair (58, 55) is removed: 2 trades, buyers pay 58, sellers receive 55; $2 \cdot 3 = 6$ stays with the market maker. The lost surplus is that of the third trade, $58 - 55 = 3$.

5.10 Further reading

- V. Krishna, *Auction Theory* (2010): the basic textbook that gives the standard, detailed proofs of all the results in this chapter (equilibrium bids, revenue equivalence, Myerson's auction, common value).
- T. Roughgarden, *Twenty Lectures on Algorithmic Game Theory* (2016), Lectures 2–5: covers the Vickrey auction, Myerson's lemma, and revenue maximization from a computer science viewpoint, briefly and clearly.
- D. Easley and J. Kleinberg, *Networks, Crowds, and Markets* (2010), Chapter 9: presents the auction formats and the winner's curse with little mathematics and intuitive examples.
- P. Milgrom, *Putting Auction Theory to Work* (2004): describes the design of spectrum auctions through how theory was carried into practice.

5.11 References

1. L. M. Ausubel, P. Milgrom, "The lovely but lonely Vickrey auction", P. Cramton, Y. Shoham, R. Steinberg (eds.), *Combinatorial Auctions*, MIT Press, 2006.
2. R. Day, P. Milgrom, "Core-selecting package auctions", *International Journal of Game Theory* 36(3–4), 393–407, 2008.
3. J. Bulow, P. Klemperer, "Auctions versus negotiations", *American Economic Review* 86(1), 180–194, 1996.
4. E. C. Capen, R. V. Clapp, W. M. Campbell, "Competitive bidding in high-risk situations", *Journal of Petroleum Technology* 23(6), 641–653, 1971.
5. D. Easley, J. Kleinberg, *Networks, Crowds, and Markets*, Cambridge University Press, 2010 (Chapter 9: Auctions).
6. J. H. Kagel, R. M. Harstad, D. Levin, "Information impact and allocation rules in auctions with affiliated private values: a laboratory study", *Econometrica* 55(6), 1275–1304, 1987.
7. A. R. Karlin, Y. Peres, *Game Theory, Alive*, American Mathematical Society, 2017 (auction chapters).
8. V. Krishna, *Auction Theory*, 2nd ed., Academic Press, 2010.
9. R. P. McAfee, "A dominant strategy double auction", *Journal of Economic Theory* 56(2), 434–450, 1992.

10. P. Milgrom, *Putting Auction Theory to Work*, Cambridge University Press, 2004.

11. R. B. Myerson, "Optimal auction design", *Mathematics of Operations Research* 6(1), 58–73, 1981.

12. R. B. Myerson, M. A. Satterthwaite, "Efficient mechanisms for bilateral trading", *Journal of Economic Theory* 29(2), 265–281, 1983.

13. N. Nisan, T. Roughgarden, É. Tardos, V. Vazirani (eds.), *Algorithmic Game Theory*, Cambridge University Press, 2007 (Chapters 9 and 13).

14. J. G. Riley, W. F. Samuelson, "Optimal auctions", *American Economic Review* 71(3), 381–392, 1981.

15. M. H. Rothkopf, T. J. Teisberg, E. P. Kahn, "Why are Vickrey auctions rare?", *Journal of Political Economy* 98(1), 94–109, 1990.

16. T. Roughgarden, *Twenty Lectures on Algorithmic Game Theory*, Cambridge University Press, 2016.

17. W. Vickrey, "Counterspeculation, auctions, and competitive sealed tenders", *Journal of Finance* 16(1), 8–37, 1961.

PART II · CHAPTER 6

Online Platforms

A marketplace, a search engine, or a ride-hailing app produces very little on its own. Its value comes from giving the people it brings together access to one another. The platform's rules (who joins, who sees whom, who pays what) are therefore mechanisms, and the users are strategic players under those rules. These notes start with two-sided markets and move on to network effects and critical mass, the question of which side is subsidized and why, GSP and VCG auctions for search ads, budgeted online allocation, reputation systems, and platform competition. In every topic a numerical example is worked through with all intermediate steps.

LEARNING OUTCOMES

Model a platform
Describe a two-sided market, cross-side network effects, and the platform as a mechanism designer.

Compute the critical mass
Find the equilibria, the threshold, and the "tipping" dynamics in a simple adoption model.

Explain the price structure
Show, in a two-sided numerical example, which side is subsidized and why.

Ad auctions
Compute GSP and VCG payments for the same bids; show why truthful bidding is not an equilibrium of GSP.

Measure reputation
Rank items with a Bayesian average; compute when fake reviews are profitable.

Discuss competition
Interpret the effects of tipping, switching costs, multi-homing, and regulation (the EU Digital Markets Act, DMA) on platform competition.

6.1 Platforms and markets

Every topic in this chapter rests on a single observation: the value of a platform depends on the number and identity of the people who use it. This observation overturns the usual market intuitions. Paying one side of the market can be profitable. Everyone's expectations can decide the fate of the market. In an auction called "second price," truthful bidding may not be the best strategy.

6.1.1 A game to start with: the sponsored search auction

When someone types "cheap flights" into a search engine, a few ads appear above the results. The positions of these ads are not chosen at random. For every search, a small auction is held among the advertisers who have bid on that keyword. Advertisers bid **per click**: "if a user clicks my ad, I will pay at most this much." Ads are placed into three **slots** according to a score that is the product of the bid and a quality measure. The top slot is more visible and gets more clicks.

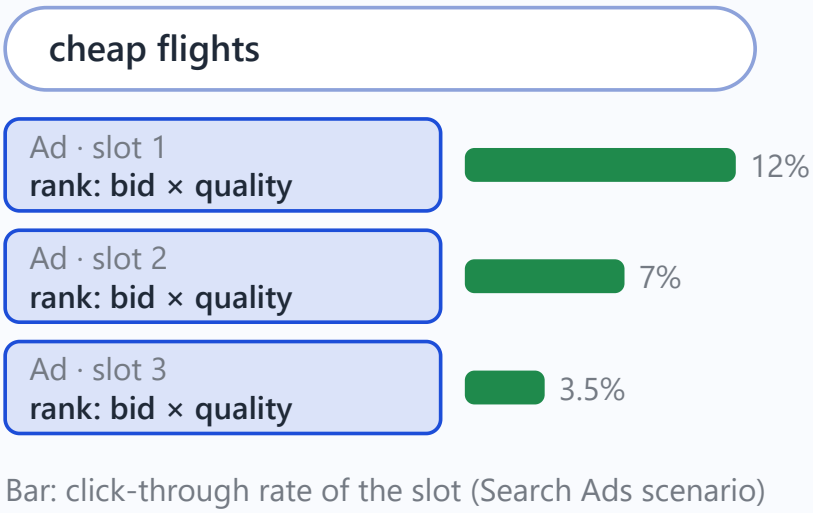


Figure 6.1: The screen of the Sponsored Search Auction game. Three ad slots are ranked by bid × quality score. The green bars are the click-through rates of the slots: 12%, 7%, and 3.5% in the game's "Search Ads" scenario.

Figure 6.1 shows the game's screen. The game asks two simple questions. First: if a click earns the advertiser 8 TL, is bidding 8 TL the best choice? Second: does the highest bidder always get the top slot? The answer to both questions is "no." Most of these notes explain why. The answer to the first question comes from the structure of the GSP auction and is shown with numbers in Section 6.4.11. The answer to the second comes from the quality score and is computed in Section 6.4.10.

To answer these questions, first recall the single-item second-price auction. In this auction (the Vickrey auction), the highest bidder wins the item but pays the second-highest bid rather than their own. Under this rule, truthful bidding, that is, bidding exactly the item's value to the bidder, is a

dominant strategy: whatever the others bid, truthful bidding never gives a worse outcome. The reason is that the bid does not determine the payment; it only determines whether you win. A bidder who bids above their value also wins in cases where the payment exceeds the value, and so incurs a loss. A bidder who bids below their value loses in cases where they could have made a profit. The GSP rule used in sponsored search tries to extend this idea to several slots, but the extension does not preserve truthfulness.

6.1.2 What is an online platform?

Definition 6.1

An **online platform** is a digital intermediary that brings together users, sellers, and service providers. Value comes less from the platform itself than from the access the sides have to one another. Such a structure is called a **two-sided market**.

The second sentence of the definition is important. A bakery produces bread, and the value of the bread does not depend on how many customers the bakery has. A marketplace, by contrast, produces nothing itself. Its value to a buyer depends on how many sellers and how much variety of goods are there. Its value to a seller depends on how many buyers are there. The real service the platform sells is lowering the cost for the two groups of finding and matching with each other.

Nor is a platform merely a meeting place. It sets the rules: who may join, who sees whom, who pays what. A marketplace decides in what order sellers are listed. A search engine decides which ad goes into which slot. A ride-hailing app decides which driver goes to which passenger, and at what price. Together, all these rules form a **mechanism**: a rule that produces an outcome and payments from the information and choices reported by the participants.

Type	Side 1	Side 2	Example
E-commerce marketplace	buyers	sellers	Trendyol, Hepsiburada, Amazon
Sharing economy	guest / rider	host / driver	Airbnb, BlaBlaCar, Uber
Social media	users	advertisers, content creators	Instagram, TikTok, Facebook
Search engine	searchers	advertisers	Google, Bing

Table 6.1: Types of platforms and their two sides.

Table 6.1 shows four common types of platforms. Every row has two distinct groups of users, and each group benefits from the presence of the other. In social media the second side consists of two groups: advertisers buy users' attention, while content creators draw users to the platform. In a search engine the searcher pays nothing at all; all of the revenue comes from advertisers. This asymmetry is no coincidence; its cause is computed in Section 6.3.

6.1.3 Two-sided markets and cross-side network effects

What separates a two-sided market from a one-sided one is that the sides add value for each other. This link is defined as follows.

Definition 6.2

Cross-side network effect: the number of participants on one side determines the value obtained by the other side. In a marketplace, many sellers mean many choices for a buyer, and many buyers mean many customers for a seller. This effect is also called an **indirect network effect**.

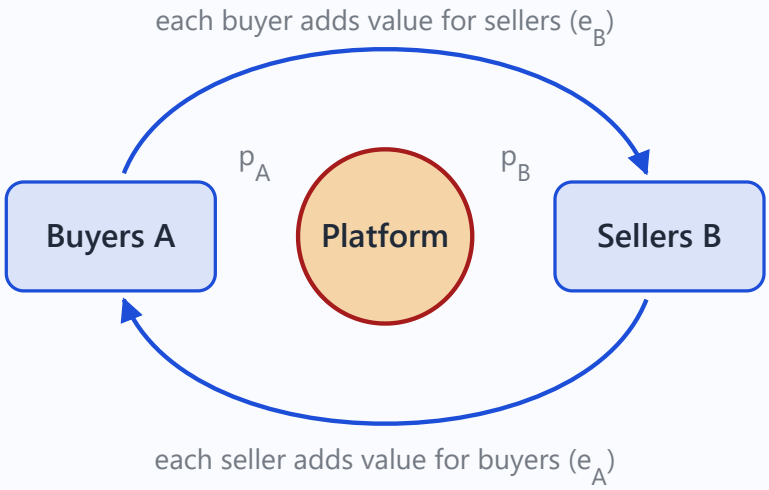


Figure 6.2: A two-sided market. The platform charges buyers p_A and sellers p_B. The upper arrow shows the value each buyer contributes to sellers (e_B); the lower arrow shows the value each seller contributes to buyers (e_A).

Figure 6.2 introduces the notation used in the rest of these notes. Side A is the buyers and side B is the sellers. p_A and p_B are the fees the two sides pay the platform. e_B measures the value a buyer contributes to *sellers*; that is, how much side B cares about side A. e_A is likewise the value a seller contributes to buyers. In this notation the subscript names the side that *receives* the value. This detail matters in the pricing rule.

The structure in the figure has three consequences.

- **Cross-side network effect.** The number of participants on one side determines the value obtained by the other side. When the platform grows one side, it automatically draws in the other.
- **Price structure.** The platform chooses not only the total price level but also how the fee is split between the sides: (p_A, p_B). Two marketplaces may charge the same total fee, but in one the buyer pays and in the other the seller pays. The outcomes can be very different.

- **The chicken-and-egg problem.** If there are no sellers, buyers do not come; if there are no buyers, sellers do not come. A new platform starts out empty, and each side waits for the other to arrive. To break the vicious circle, one of the sides must be "drawn in" first with free use, discounts, or direct payments.

The chicken-and-egg problem gives rise to the two main questions of this chapter. First: under what conditions does a platform grow, and under what conditions does it collapse? This is the critical mass question (Section 6.2). Second: which side should be drawn in, that is, subsidized? This is the price structure question (Section 6.3).

6.1.4 The platform is a mechanism, the users are players

Mechanism design is the art of choosing game rules that will produce a desired outcome. Platforms are one of the largest application areas of this art. Platform design (fee, ranking, and rating rules) steers users' strategic behavior. Users in turn choose strategies according to incentives. The behavior that emerges is the **Nash equilibrium** under the rules: the situation in which no user wants to change their own behavior given the behavior of the others.

This has a practical consequence: a rule should be evaluated not on the assumption that users will behave "in good faith," but on the assumption that they will pursue their own interests. When a platform introduces an incentive, users work out how best to use it. Sometimes this calculation produces the behavior the platform wanted; sometimes it produces an unexpected exploit.

Platform incentive	Users' strategic response
Bonus: extra pay for drivers in busy areas (Uber)	Fake reviews and rating inflation
Discounts, coupons, free trials	Opening multiple accounts, bonus abuse
Ranking and recommendation algorithm → visibility	Manipulating the algorithm (click farms)
Ratings and badges (trusted seller)	Retaliatory bad reviews

Table 6.2: Platform incentives and users' strategic responses.

Table 6.2 puts two columns side by side. Each tool on the left is designed to reward a particular behavior. Each behavior on the right is an abuse of a tool. For example, a free trial is offered to attract new users, but one person can open many accounts and use the trial again and again. A rating and badge system is introduced to promote trustworthy sellers, but if ratings can be bought, the badge measures not trustworthiness but willingness to cheat.

The lesson of the table is this: the right incentive brings cooperation and efficiency, and the wrong incentive brings manipulation and loss of trust. The platform designer's job is to find rules under which users, by following their own interests, reach the outcome that the platform and society want. The rest of these notes studies this job in four areas: participation (network effects), price structure, ad auctions, and reputation systems.

Common mistake

"A platform is only an intermediary; users' behavior is independent of the platform." This is wrong. Users' behavior is an equilibrium under the rules the platform sets. The same users behave differently under a different ranking or fee rule. For example, if the average rating determines the ranking, the payoff from writing fake reviews rises; if the ranking pulls the scores of items with few reviews toward the average, that payoff falls (Section 6.5.3).

6.2 Network effects and critical mass

If the value of a product rises with the number of people who use it, the question "how many people will use it?" is as important as the product itself. After the types of network effects, a simple one-parameter model explains three phenomena. There are several possible outcomes for the same platform. Expectations determine which of these outcomes is reached. A small difference in the starting point can turn into a large difference in the outcome.

6.2.1 Types of network effects

Definition 6.3

If the value that using a product or platform gives a person depends on the number of other people using it, this link is called a **network effect**. If an additional user raises the value, the network effect is *positive*; if it lowers the value, it is *negative*.

Network effects fall into four categories.

Direct (same side)
: Value comes from the number of users on the same side. The more of a person's friends use WhatsApp, the more valuable the app is to that person. Telephones, e-mail, and messaging apps are of this type.

Indirect / cross-side
: Value comes from the number of users on the other side. On Uber, many drivers mean a short wait; a short wait attracts many riders; and many riders mean more work for drivers. For a rider, what matters is not the other riders but the drivers.

Negative
: Crowding sometimes lowers value: congestion, information pollution, long queues. In a two-sided market, sellers on the same side are rivals; for a seller, a new seller means that buyers will have to be shared.

Local

> Value comes not from all users but from the immediate neighborhood. For a rider in Istanbul, drivers in Ankara are irrelevant. If a whole group of friends moves to another messaging app, the app's total user count means nothing to that group. This is why competition is won locally: city by city, group by group.

A positive network effect sets up a **feedback loop**. A growing platform becomes more valuable, and because it is more valuable it grows even more. A shrinking platform loses value, and because it loses value it shrinks even more. In such a loop the market is expected to "tip" in one direction: either the platform reaches critical mass and grows, or it fails to reach it and collapses. Intermediate states are unstable. Making this idea precise requires a model.

6.2.2 A simple adoption model

The model below is a simplified version of the network effect model of Easley and Kleinberg (2010, Chapter 17). Its aim is not realism but to show the essence of the dynamics with the fewest possible assumptions.

Definition 6.4

Adoption model. There are $N = 1000$ potential users. The **type** θ_i of user i is a number between 0 and 1 that indicates how much they care about the platform, and it is uniformly distributed on this interval: $\theta_i \sim U[0, 1]$. If a fraction x of the population joins, the payoff user i gets from joining is

$$\theta_i \cdot x - p$$

Here $p = 0.16$ is the cost of joining (a fee or the effort of joining). A user who does not join gets a payoff of 0.

Every part of the definition has a meaning. The term $\theta_i \cdot x$ is the network effect: the value of the platform rises in proportion to the fraction x of participants. If nobody is there ($x = 0$), the platform gives no one any value. θ_i is personal interest: the same crowd is worth a lot to someone who cares a great deal about the platform (θ near 1) and little to someone who cares little (θ near 0). p is a fixed cost that is the same for everyone.

The notation $\theta \sim U[0, 1]$ means "types are spread with equal probability between 0 and 1." Its practical meaning is simple: for any threshold a, the fraction of users whose type exceeds a is $1 - a$. For example, those with a type above 0.64 make up 36% of the population; among 1000 people, that is 360.

Who joins? A user joins if their payoff is nonnegative:

$$\theta_i \cdot x \geq p \iff \theta_i \geq p / x$$

In the second step both sides are divided by x; since x is positive, the direction of the inequality does not change. This inequality gives the lowest type needed to join: $\theta^* = p / x$. Those whose type is above this threshold join. By the uniform distribution, their fraction is $1 - p/x$.

There is a subtle but important point here. When deciding whether to join, the user does not yet know x, because x depends on everyone's decision. The user decides according to an **expectation**: "I think a fraction x of the population will join." The fraction that actually joins under this expectation is given by the function

$$F(x) = 1 - p/x \quad (\mathrm{F(x)} = 0 \text{ if } x \le p)$$

The case in parentheses is explained as follows. If $x \le p$, the threshold $p/x \ge 1$. Nobody has a type above 1, so nobody joins. If the expected crowd is too small, even the user who cares most about the platform sees no value that covers the cost.

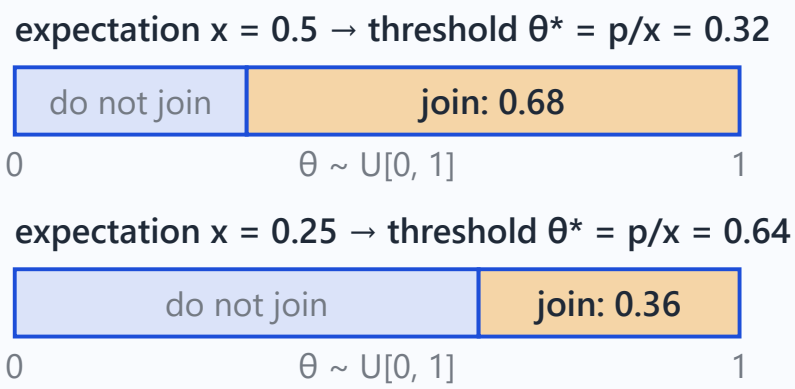


Figure 6.3: The expectation determines the participation threshold. With expectation $x = 0.5$ the threshold is $\theta^* = 0.32$ and 0.68 of the population joins. When the expectation falls to $x = 0.25$, the threshold rises to 0.64 and only 0.36 join.

Example 6.1

1. **Expectation x = 0.5.** Suppose everyone thinks half the population (500 people) will join. The threshold is $\theta^* = p/x = 0.16/0.5 = 0.32$. Those with a type above 0.32 join: $F(0.5) = 1 - 0.32 = 0.68$, that is, 680 people.
2. **Expectation x = 0.25.** Suppose everyone thinks 250 people will join. The threshold rises to $0.16/0.25 = 0.64$. The fraction that joins is $F(0.25) = 1 - 0.64 = 0.36$, that is, 360 people.
3. **Comparison.** When the expectation is halved, the threshold doubles. Because the platform's value falls as the crowd shrinks, only those who care more about the platform join.

4. **Expectation versus realization.** In both cases the realized participation differs from the expectation: 0.68 joined when 0.5 was expected, and 0.36 joined when 0.25 was expected. The gap between expectation and realization changes the expectation of the next period.

Figure 6.3 shows these two cases with bars. Each bar lies on the type axis between 0 and 1. The orange part is those who join and the blue part is those who do not. As the expectation falls, the boundary moves to the right and the orange part shrinks.

6.2.3 Self-fulfilling expectations: equilibria

In both cases of Example 6.1 the expectation turned out wrong. The case in which the expectation turns out right, that is, exactly as many people join as everyone expected, is of special importance.

Definition 6.5

In the adoption model, an **equilibrium** is a **self-fulfilling expectation**: everyone expects a fraction x to join, and under this expectation a fraction x really does join. Formally, an equilibrium is a solution of the equation $F(x) = x$.

This definition is the counterpart of Nash equilibrium in this model. In equilibrium no user wants to change their decision: those who join have a nonnegative payoff, and those who do not join would have a negative payoff if they did. Since the expectation is correct, nobody has been wrong.

Example 6.2

1. **Set up the equation.** For $x > p$, the condition $F(x) = x$ becomes $1 - 0.16/x = x$.
2. **Clear the fraction.** Multiply both sides by x (x is positive): $x - 0.16 = x^2$. Collecting the terms on one side gives $x^2 - x + 0.16 = 0$.
3. **Solve the quadratic.** The roots of $ax^2 + bx + c = 0$ are $x = (-b \pm \sqrt{(b^2 - 4ac)}) / 2a$. Here $a = 1$, $b = -1$, $c = 0.16$. The expression under the root (the discriminant) is $1 - 4 \cdot 0.16 = 1 - 0.64 = 0.36$, whose square root is 0.6. The roots are $(1 - 0.6)/2 = 0.2$ and $(1 + 0.6)/2 = 0.8$.
4. **Check.** $F(0.2) = 1 - 0.16/0.2 = 1 - 0.8 = 0.2$ ✓ and $F(0.8) = 1 - 0.16/0.8 = 1 - 0.2 = 0.8$ ✓.
5. **Do not forget zero.** $x = 0$ is also an equilibrium: if it is expected that nobody will join, then $x \le p$, so $F(0) = 0$ and indeed nobody joins.

The model has three equilibria: $x = 0$ (nobody), $x = 0.2$ (200 users), and $x = 0.8$ (800 users).

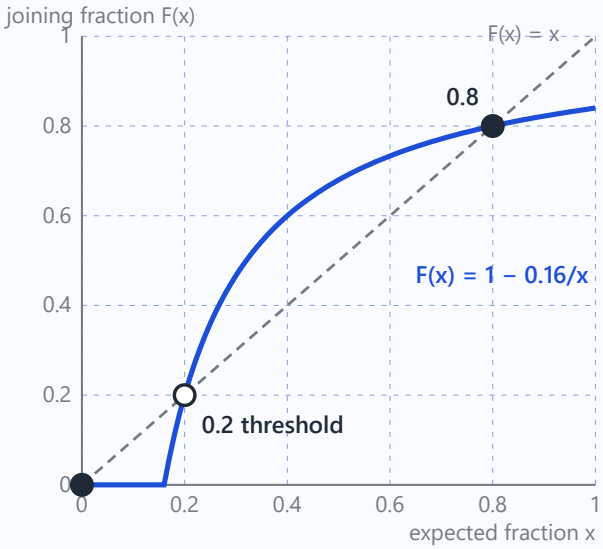


Figure 6.4: The adoption curve $F(x) = 1 - 0.16/x$ (blue) and the 45° line $F(x) = x$ (dashed). The intersections are the equilibria: 0 and 0.8 (filled dots) and 0.2 (hollow dot). For $x \le 0.16$ the curve is at zero.

Figure 6.4 shows the equilibria graphically. The horizontal axis is the expected fraction x, and the vertical axis is the fraction that actually joins, $F(x)$. At every point on the dashed 45° line, expectation and realization are equal. The equilibria are the points where the blue curve crosses the dashed line. The curve is flat up to 0.16, then rises quickly and approaches 1 without reaching it.

There are two kinds of points in the graph. The filled dots (0 and 0.8) are **stable** equilibria, and the hollow dot (0.2) is an **unstable** equilibrium. An unstable equilibrium means this: if the expectation deviates slightly from this point, the dynamics push it away from the equilibrium.

That a platform can be both successful (800 users) and a failure (0 users) under the same conditions is the most important lesson of this model. The product, the price, and the users are the same; the only thing that decides the outcome is expectation. This is the platform version of the **multiple equilibria** problem in game theory: as in coordination games, everyone acts according to what the others will do, and both “everyone comes” and “nobody comes” are self-fulfilling.

6.2.4 Tipping step by step

Defining an equilibrium does not say how one gets there. For that we assume a simple dynamic over time: each period, users take the participation fraction they observed in the previous period as this period’s expectation. Those who join this period in turn determine the next period’s expectation:

$$x_{t+1} = F(x_t)$$

Here x_t is the expected fraction in period t, and x_{t+1} is the fraction that results from that expectation and becomes the expectation of the next period. The dynamics start from an initial expectation x_0.

Example 6.3

Starting above the threshold: $x_0 = 0.25$. Suppose the platform has created the impression that 250 users are expected at the start.

1. $t = 0 \to 1$: threshold $\theta \geq 0.16/0.25 = 0.64$; fraction joining $x_1 = 1 - 0.64 = 0.36$.
2. $t = 1 \to 2$: participation exceeded the expectation (0.36 > 0.25). With the new expectation 0.36 the threshold is $0.16/0.36 \approx 0.444$; $x_2 = 1 - 0.444 = 0.556$.
3. $t = 2 \to 3$: the feedback continues: $x_3 = 1 - 0.16/0.556 = 0.712$.
4. $t = 3 \to 4$: $x_4 = 1 - 0.16/0.712 = 0.775$. The increments begin to shrink: 0.11, 0.20, 0.16, 0.06.
5. $t = 4 \to 6$: $x_5 = 0.794$, $x_6 \approx 0.798$. The path approaches the stable equilibrium 0.8. The market has tipped in the platform's favor: about 800 users.

Starting below the threshold: $x_0 = 0.19$. Suppose the initial expectation is only slightly below the critical threshold.

1. $x_1 = 1 - 0.16/0.19 = 1 - 0.842 = 0.158$. Participation fell *short of* the expectation.
2. Since $x_1 = 0.158 \leq p = 0.16$, we get $x_2 = 0$: nobody joins and the platform collapses.

t	$x_0 = 0.25$	$x_0 = 0.19$
0	0.25	0.19
1	0.36	0.158
2	0.556	0
3	0.712	0
4	0.775	0
5	0.794	0
6	0.798	0

Table 6.3: The dynamics $x_{t+1} = F(x_t)$ with $p = 0.16$ for two initial expectations.

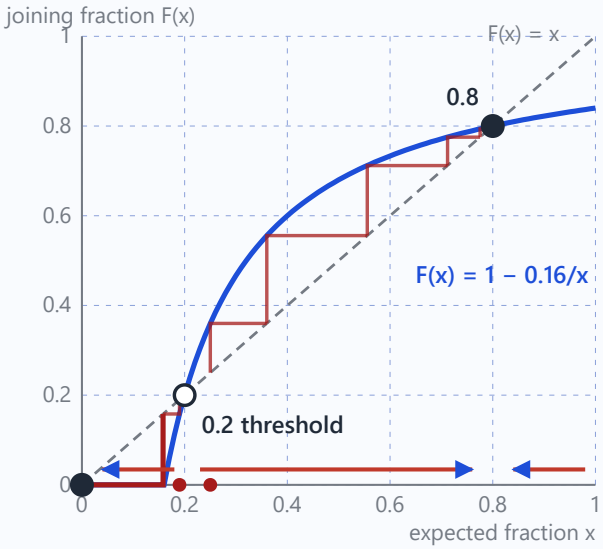


Figure 6.5: A "cobweb" plot of the tipping dynamics. The red steps start at $x_0 = 0.25$ and climb to 0.8; the short path starting at $x_0 = 0.19$ falls to 0. The arrows at the bottom show the direction of the dynamics: to the right of 0.2 they point right (toward 0.8), to the left of 0.2 they point left (toward 0).

Figure 6.5 shows one way to follow the dynamics on the graph. This drawing is called a **cobweb plot** and is read as follows. Start at x_t on the horizontal axis. Go vertically up to the curve; the height of the curve is $x_{t+1} = F(x_t)$. Then go horizontally to the 45° line, which "carries" the new value over to the horizontal axis. From there go vertically to the curve again, and so on. If the curve is above the line, the steps move right; if it is below, they move left.

The consequence of this rule is clear in the graph. Between 0.2 and 0.8 the curve lies above the line: in every period participation exceeds the expectation and the fraction grows. To the right of 0.8 the curve is below the line: the fraction shrinks and returns to 0.8. To the left of 0.2 the curve is also below the line: the fraction shrinks and falls to 0. So 0.8 and 0 attract paths from both sides, while 0.2 pushes them away in both directions.

Table 6.3 gives a striking comparison. The difference between the two starting points is only 0.06 (60 users). Yet one goes to 800 users and the other to zero. This is the mathematical reason for the "all or nothing" behavior of platforms.

Definition 6.6

The unstable equilibrium of the adoption model is called the **critical mass**. If the initial expectation is above the critical mass, the dynamics go to the large stable equilibrium; if it is below, they go to zero. The market's spontaneous movement to one of these two extremes is called **tipping**.

In this example the critical mass is 0.2, that is, 200 users. If the expectation is above it, the market goes to 0.8; if below, to 0.

6.2.5 Which equilibrium is stable?

Stability is easy to see in the graph, and the same conclusion can be reached by computation. Consider a small deviation near an equilibrium: $x_t = x^* + d$. One period later the deviation is approximately $F'(x^*) \cdot d$. Here $F'(x^*)$ is the slope of the curve at the equilibrium point. If the slope has absolute value less than 1, the deviation shrinks every period and the equilibrium is stable. If it is greater than 1, the deviation grows and the equilibrium is unstable.

For $F(x) = 1 - p/x$ the slope is $F'(x) = p/x^2$. This follows from writing the term p/x as $p \cdot x^{-1}$, whose derivative is $-p \cdot x^{-2}$, together with the minus sign in front.

- $x^* = 0.8$: $F'(0.8) = 0.16/0.64 = 0.25 < 1$. The deviation falls to a quarter each period: stable. For example, starting from $x = 0.81$ gives $F(0.81) \approx 0.802$; the deviation of 0.01 has fallen to 0.002.
- $x^* = 0.2$: $F'(0.2) = 0.16/0.04 = 4 > 1$. The deviation roughly quadruples each period: unstable. For example, starting from $x = 0.21$ gives $F(0.21) \approx 0.238$; the deviation of 0.01 has grown to about 0.038.
- $x^* = 0$: immediately around 0, F is zero (the curve is flat); the slope is 0. A small expectation returns to zero at once: stable.

This computation explains why the filled and hollow dots in Figure 6.4 are drawn as they are. If the curve crosses the 45° line *steeply* (slope greater than 1), the equilibrium is unstable; if it crosses *shallowly* (slope less than 1), it is stable.

6.2.6 Fees and thresholds: design lessons

The model depends on a single parameter, p. The platform can influence this parameter: it can lower the fee, make joining easier, or give the first users a bonus. When p changes, the equilibria change too. For every p the same equation is solved: $x^2 - x + p = 0$.

p	equilibria	threshold (critical mass)
0.09	0, 0.1, 0.9	0.1
0.16	0, 0.2, 0.8	0.2
0.21	0, 0.3, 0.7	0.3
0.25	0 and 0.5 (tangent)	0.5
0.3	only 0	—

Table 6.4: Equilibria and critical mass as a function of the cost of joining p.

Example 6.4

p = 0.21. The equation is $x^2 - x + 0.21 = 0$. The discriminant is $1 - 0.84 = 0.16$, with square root 0.4. The roots are $(1 - 0.4)/2 = 0.3$ and $(1 + 0.4)/2 = 0.7$. Check: $F(0.3) = 1 - 0.21/0.3 = 1 - 0.7 = 0.3$ ✓, $F(0.7) = 1 - 0.21/0.7 = 1 - 0.3 = 0.7$ ✓.

p = 0.25. The discriminant is $1 - 1 = 0$; the single root is $x = 1/2 = 0.5$. At this point the curve only touches the 45° line and does not cross it (tangency). On both sides of 0.5 the curve is below the line. So starting below 0.5 the path goes to 0. Starting above, participation decreases and approaches 0.5. This equilibrium is stable from one direction only: a small drop makes the market collapse.

p = 0.3. The discriminant is $1 - 1.2 = -0.2 < 0$: there is no real root. The curve is always below the 45° line; whatever expectation one starts from, participation shrinks and goes to 0. The market never forms.

Table 6.4 and Example 6.4 show two general results. First: as the fee falls, the threshold falls and the number of users at the large equilibrium rises. When p drops from 0.21 to 0.09, the critical mass falls from 300 to 100 users. Second: for the equation $x^2 - x + p = 0$ to have a real root, the discriminant must satisfy $1 - 4p \geq 0$, that is, $p \leq 0.25$. If $p > 0.25$, the market never forms. The critical mass and the large equilibrium always add up to 1 (0.2 + 0.8, 0.3 + 0.7), because in a quadratic the sum of the roots is $-b/a = 1$. As the threshold rises, the large equilibrium falls by the same amount.

These results explain why platforms behave unusually at the start. There are four common strategies:

Initial subsidy
: A free period or a bonus reduces p and lowers the critical mass. Once the crowd has formed, the platform can raise the fee, because the large equilibrium is stable and a small fee increase does not destroy it.

Seeding
: The platform itself supplies the first content, sellers, or drivers. The actual participation x is then greater than zero before users' participation even begins.

Expectation management
: The perception that “everyone is here” moves the initial expectation x_0 above the threshold. The model shows that the expectation itself is an input; advertising and promotion change this input.

Standalone value
: One starts with a tool that is useful even without the network and builds the network later. For example, an app that began as a photo-editing tool attracts users even without a network effect; when a sharing feature is added, the crowd is ready. In the model this means adding a term to the payoff that does not depend on x; then $x = 0$ is no longer an equilibrium.

History shows both directions. WhatsApp’s value grew exponentially as users increased. Friendster, on the other hand, collapsed in the opposite direction through the same feedback loop once it began losing users. The mechanism at work is the same in both cases; the difference is which side of the threshold the expectation was on.

6.2.7 Common mistakes with network effects

Common mistake

Forgetting the expectation. Writing $1 - p$ when computing $F(x)$, that is, forgetting that participation depends on x. For $p = 0.16$ and $x = 0.4$ the correct answer is $F(0.4) = 1 - 0.16/0.4 = 1 - 0.4 = 0.6$, not 0.84.

Confusing the threshold type with the fraction who join. In the same example, 0.4 is the type at the joining threshold ($\theta^* = p/x$), not the fraction who join. The fraction who join is the part above the threshold: $1 - \theta^*$.

Skipping the zero equilibrium. The equation $x^2 - x + p = 0$ gives only the positive equilibria. Because the equation was multiplied by x when it was set up, $x = 0$ must be checked separately; the situation in which nobody joins is always an equilibrium.

Mistaking the unstable equilibrium for a "middle way." Although 0.2 is an equilibrium, the market does not stay there. Since real expectations are never exactly 0.2, 0.2 is not a target but a boundary.

6.3 Platform pricing

In a one-sided market the pricing question is simple: at what price should the product be sold? On a two-sided platform the question splits in two: how much should be charged in total, and how should that total be divided between the sides? The answer to the second question is found with a numerical model and is surprising at first glance: a profit-maximizing platform pays one of the sides.

6.3.1 Principles of two-sided pricing

According to Rochet and Tirole (2003), besides the overall price level a platform also decides the **price structure**: which side pays the fee, and how much? For a one-sided firm there is no such thing as a "price structure." In a two-sided market, however, the structure matters as much as the overall level, because one side's fee changes not only that side's participation but also, through the cross-side network effect, the other side's participation.

The basic principle can be summarized in two points:

- The side that adds more value to the other side, or that is more price-sensitive, is charged less, or even **subsidized**. A subsidy means a negative price: the platform pays every participant on that side.
- This loss is covered by the fee collected from the other side, which the subsidized side draws to the platform.

There are many everyday examples of this principle:

Search and social media
: Users use the service for free and advertisers pay. Users' attention is valuable to advertisers; charging users would reduce that attention.

Credit cards
: Cardholders receive points and miles, and merchants pay a commission. A cardholder's use of the card is valuable to the merchant; points increase that use.

Marketplaces
: Buyers get coupons and sellers pay a commission.

Revenue models
: Commission, membership, advertising, per-transaction fees. These models determine which side bears the fee and in what form.

Armstrong (2006) reduces this principle to a single rule in a simple linear model:

> p_A = *markup – the value an A participant adds to side B*

According to the rule, if the second term is large, the price falls below zero. The **markup** is the difference between the price the platform would charge a participant, and the cost, if there were no network effect. The logic of the rule is this: when the platform attracts a side-A participant, it gains two things. The first is the fee collected from that participant. The second is the value that participant adds to side B, because side B is willing to pay more for that value. If the second gain is large, the platform is willing to pay an A participant to attract them. The rule is derived in general form in Section 6.3.5.

6.3.2 A numerical example: the marketplace model

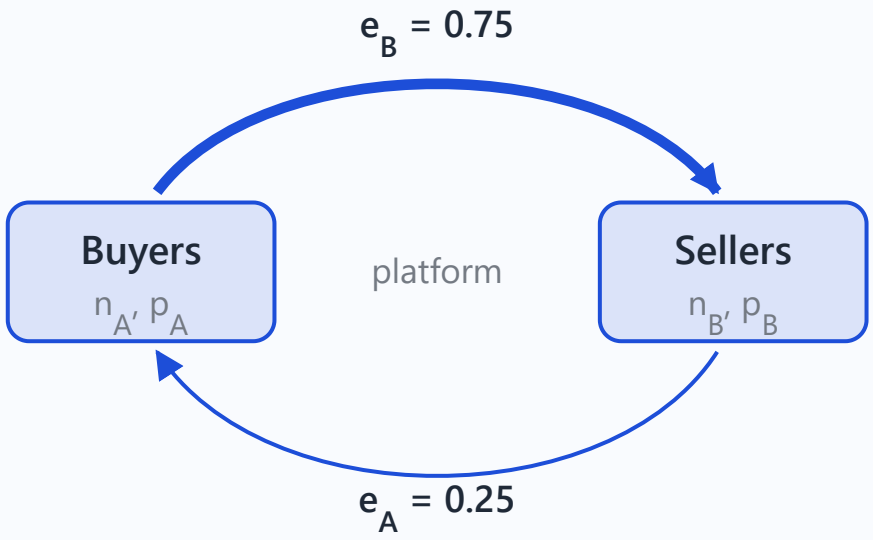


Figure 6.6: The marketplace model. Thick upper arrow: each buyer adds $e_B = 0.75$ of value for sellers. Thin lower arrow: each seller adds $e_A = 0.25$ of value for buyers.

Consider a marketplace. Let n denote the number of participants (in thousands) and p the monthly fee (in TL). Serving one more participant costs the platform nothing (marginal cost 0). For a seller, buyers are very valuable ($e_B = 0.75$); for a buyer, seller variety matters less ($e_A = 0.25$). Participation is determined by these two equations:

$$n_A = 20 + 0.25 \cdot n_B - p_A$$
$$n_B = 80 + 0.75 \cdot n_A - p_B$$

The platform's profit is the sum of the fees collected from the two sides:

$$\Pi = p_A \cdot n_A + p_B \cdot n_B$$

Reading the equations one at a time reveals the assumptions of the model. The 20 in the buyer equation is the number of buyers (in thousands) who would come even with a zero fee and no sellers at all. The term $0.25 \cdot n_B$ is the additional buyers drawn by sellers: 250 buyers per thousand sellers. The term $-p_A$ is the buyers lost to the fee: every 1 TL increase in the fee costs a thousand buyers. The seller equation has the same form, but its constant (80) and network effect coefficient (0.75) are larger. The importance sellers attach to buyers is three times the importance buyers attach to sellers.

The platform's question is: what are the profit-maximizing (p_A, p_B), and which side is subsidized?

6.3.3 The profit-maximizing prices step by step

Computing directly in terms of prices is hard, because each price changes both participation levels and the levels depend on each other. There is a way to make the computation easier: make the participation levels, rather than the prices, the decision variables. Each pair (n_A, n_B) corresponds to a single pair of prices. The platform can think "I want this many buyers and this many sellers" and read the prices that deliver those numbers from the equations.

The solution uses **partial derivatives**. The partial derivative of a function of two variables with respect to one variable is the derivative taken while the other variable is held constant; it is written with ∂. For example, the partial derivative of $n_A\, n_B$ with respect to n_A is n_B, because n_B behaves like a constant. At the peak of a function of two variables both partial derivatives are zero: at the top, the slope is zero whichever direction one steps in.

Example 6.5

1. **Make the participation levels the decision variables.** Solve the participation equations for the prices: $p_A = 20 + 0.25\, n_B - n_A$ and $p_B = 80 + 0.75\, n_A - n_B$.
2. **Substitute into the profit.** $p_A\, n_A = 20\, n_A + 0.25\, n_A n_B - n_A^2$ and $p_B\, n_B = 80\, n_B + 0.75\, n_A n_B - n_B^2$. Adding them gives

   $$\Pi = 20\, n_A + 80\, n_B + (0.25 + 0.75)\, n_A n_B - n_A^2 - n_B^2$$

   The coefficient of $n_A n_B$ is the sum of the two cross effects (here 1). This term shows that the platform can collect the value the two sides add for each other.
3. **Set the partial derivatives to zero.** With n_B held fixed, $\partial\Pi/\partial n_A = 20 + n_B - 2n_A = 0$. With n_A held fixed, $\partial\Pi/\partial n_B = 80 + n_A - 2n_B = 0$. This gives two linear equations.
4. **Solve the equations.** From the first, $n_A = (20 + n_B)/2$. Substituting into the second gives $80 + (20 + n_B)/2 - 2n_B = 0$. Multiplying both sides by 2: $160 + 20 + n_B - 4n_B = 0$, that is, $3n_B = 180$ and $n_B = 60$. Substituting back, $n_A = 80/2 = 40$.
5. **Read off the prices.** $p_A = 20 + 0.25 \cdot 60 - 40 = 20 + 15 - 40 = -5$ and $p_B = 80 + 0.75 \cdot 40 - 60 = 80 + 30 - 60 = 50$.
6. **Compute the profit.** $\Pi = (-5) \cdot 40 + 50 \cdot 60 = -200 + 3000 = 2800$.
7. **Is it a peak?** In the profit, the terms $-n_A^2$ and $-n_B^2$ give the function a "dome" shape; the coefficient of the cross term (1) is not large enough to spoil this dome (the condition: the coefficient must be less than 2). The point found is therefore a peak.

Solution: buyers are paid 5 TL per month ($p_A = -5$), and sellers pay 50 TL. The platform has 40 thousand buyers and 60 thousand sellers; the profit is 2800 thousand TL.

The interpretation of the result is the main idea of the chapter. The platform pays buyers but does not lose money. The 200 paid to buyers is small next to the 3000 collected from sellers. Sellers are willing to pay 50 TL because the platform has 40 thousand buyers, and every buyer adds value for sellers.

6.3.4 Comparing the scenarios

To see that the subsidy really is profitable, other price structures are tried. In each scenario the buyer fee is fixed and the seller fee is chosen optimally under that condition.

Scenario	e_A / e_B	p_A	p_B	n_A	n_B	Π
Optimum	0.25 / 0.75	−5	50	40	60	2800
Free for buyers	0.25 / 0.75	0	47.5	34.6	58.5	2776.9
Buyers pay 10 TL	0.25 / 0.75	10	42.5	23.8	55.4	2592.3
Effects reversed	0.75 / 0.25	25	30	40	60	2800

Table 6.5: Four scenarios in the marketplace model (participation in thousands, fees in TL, profit in thousand TL).

Example 6.6

Free for buyers ($p_A = 0$). Suppose the platform, instead of paying buyers, simply chooses not to charge them. This time the participation levels are computed from the prices.

1. The seller equation is substituted into the buyer equation: $n_A = 20 + 0.25(80 + 0.75\, n_A - p_B)$. Expanding gives $n_A = 40 + 0.1875\, n_A - 0.25\, p_B$.
2. The terms in n_A are collected on one side: $0.8125\, n_A = 40 - 0.25\, p_B$. Since 0.8125 = 13/16, $n_A = (640 - 4\, p_B)/13$.
3. The number of sellers: $n_B = 80 - p_B + 0.75\, n_A = (1520 - 16\, p_B)/13$.
4. The profit comes only from sellers: $\Pi = p_B\,(1520 - 16\, p_B)/13$. This is a downward parabola; its peak is where the derivative is zero: $1520 - 32\, p_B = 0$, that is, $p_B = 47.5$.
5. Participation: $n_A = (640 - 190)/13 = 450/13 \approx 34.6$, $n_B = (1520 - 760)/13 = 760/13 \approx 58.5$. Profit: $47.5 \cdot 760/13 = 36100/13 \approx 2776.9$.

Free access for buyers earns only 23.1 thousand TL less than the optimum, but it still falls behind: the number of buyers drops from 40 to 34.6 and sellers pay less.

Buyers pay 10 TL. The same method gives $n_A = (480 - 4\, p_B)/13$ and $n_B = (1400 - 16\, p_B)/13$. The profit is $\Pi = [10(480 - 4p_B) + p_B(1400 - 16p_B)]/13$; the derivative gives $1360 - 32\, p_B = 0$, that is, $p_B = 42.5$. Participation is $310/13 \approx 23.8$ and $720/13 \approx 55.4$, and the profit is $33700/13 \approx 2592.3$.

Table 6.5 gives three lessons at once.

- **The subsidy is profitable.** As buyers are charged, profit falls: 2800 → 2776.9 → 2592.3. If buyers are charged 10 TL, the number of buyers falls to 23.8 thousand, and sellers decline too. The 238 thousand TL collected from buyers does not make up for the loss on the seller side.
- **Why?** An extra buyer adds $0.75 \cdot 60 = 45$ TL of value for sellers, and the platform recovers this through p_B. The buyer's markup is 40. Since $45 > 40$, the buyer is subsidized: $40 - 45 = -5$.
- **Counterfactual.** If the cross effects were reversed, that is, if buyers added little value for sellers ($e_A = 0.75$, $e_B = 0.25$), the subsidy would disappear. The profit expression does not change (the cross term is still $0.75 + 0.25 = 1$), so participation is still 40 and 60. But the prices change: $p_A = 20 + 0.75 \cdot 60 - 40 = 25$, $p_B = 80 + 0.25 \cdot 40 - 60 = 30$. This time buyers are willing to pay, because 60 thousand sellers add a great deal of value for them.

The last row shows a subtlety. Profit and participation depend only on the *sum* of the cross effects. The price structure, however, depends on the *distribution* of the effects. The total value the platform collects does not change; what changes is which side it is collected from. The script in the presentation scans the grid of (p_A, p_B) in steps of 0.5 TL and finds the highest profit at the point $(-5, 50)$. This is a numerical check of the solution found with derivatives.

6.3.5 The general rule: markup equals the value a participant brings

The prices in the numerical example obey the rule $p_A = n_A - e_B \cdot n_B = 40 - 0.75 \cdot 60 = -5$. This is no coincidence. The rule holds in every linear model of the same form.

Theorem 6.1 (Pricing rule in the linear model)

Let participation be given by $n_A = a + e_A\, n_B - p_A$ and $n_B = b + e_B\, n_A - p_B$, with marginal cost 0 and $e_A + e_B < 2$. The profit-maximizing prices are:

$$p_A = n_A - e_B \cdot n_B,\; p_B = n_B - e_A \cdot n_A$$

Here n_A is the markup on side A, and $e_B \cdot n_B$ is the total value an A participant adds to side B.

Proof. The steps of the example are repeated with general numbers. The prices are written in terms of participation as $p_A = a + e_A\, n_B - n_A$ and $p_B = b + e_B\, n_A - n_B$. Substituting into the profit expression gives $\Pi = a\, n_A + b\, n_B + (e_A + e_B)\, n_A n_B - n_A^2 - n_B^2$.

The partial derivative with respect to n_A is set to zero: $a + (e_A + e_B)\, n_B - 2n_A = 0$. This gives $a = 2n_A - (e_A + e_B)\, n_B$.

Substitute this into the formula for p_A: $p_A = 2n_A - (e_A + e_B)\, n_B + e_A\, n_B - n_A$. The $e_A\, n_B$ terms cancel, leaving $p_A = n_A - e_B\, n_B$. The computation for side B is symmetric.

Finally, we show that the point found is a peak. The second derivatives of the profit are -2 and -2, and the cross derivative is $e = e_A + e_B$. For a function of two variables, if $(-2)(-2) - e^2 > 0$ and the diagonal terms are negative, the stationary point is a peak. This condition is equivalent to $e < 2$, which is the assumption of the theorem. □

The reason the rule contains a "markup" term is as follows. Without network effects ($e_A = e_B = 0$), side A would reduce to a one-sided monopoly problem: with demand $n_A = a - p_A$ the profit is $p_A(a - p_A)$, and the best price is $a/2$; at this price $n_A = a/2$. That is, with linear demand the monopoly price equals the number of participants. The term n_A in the rule is this monopoly price. The second term is the correction for the network effect.

Example 6.7

Theorem 6.1 is applied to the scenarios in the table.

1. **Optimum ($e_A = 0.25$, $e_B = 0.75$).** $p_A = 40 - 0.75 \cdot 60 = -5$ ✓; $p_B = 60 - 0.25 \cdot 40 = 50$ ✓.
2. **Effects reversed.** $p_A = 40 - 0.25 \cdot 60 = 25$ ✓; $p_B = 60 - 0.75 \cdot 40 = 30$ ✓.
3. **Equal effects ($e_A = e_B = 0.5$).** Since the sum is still 1, participation is 40 and 60. $p_A = 40 - 0.5 \cdot 60 = 10$, $p_B = 60 - 0.5 \cdot 40 = 40$. The profit is $400 + 2400 = 2800$. In this case there is no subsidy: the value a buyer adds for sellers (30) is smaller than the markup (40).

The rule answers the "Why?" question in Table 6.5 in one line. The criterion for a subsidy is not that a side is crowded, that it brings in revenue, or that it consists of sellers. The criterion is whether the value one participant on that side adds to the other side exceeds the markup.

6.3.6 Code trace: participation()

In Example 6.5 the participation levels were found as the joint solution of two equations. In reality users do not solve a system of equations; each side reacts to the other's current size. The short function below imitates this process: the prices are fixed, and the sides react to each other in turn.

```
def participation(pA, pB, rounds=8):
  nA, nB = 0, 0
  for t in range(rounds):
    nA = 20 + 0.25 * nB - pA   # buyers
    nB = 80 + 0.75 * nA - pB   # sellers
  return nA, nB
```

The logic of the code is as follows:

- Initially the platform is empty: `nA, nB = 0, 0`.
- In each round the buyers react first, to the *current* number of sellers: `nA = 20 + 0.25 * nB - pA`.

- Then the sellers react to the *new* number of buyers: `nB = 80 + 0.75 * nA - pB`. The second line uses the `nA` that the first line updated in the same round.
- After the loop has repeated `rounds` times, the final participation levels are returned. The default of 8 rounds is enough for this example.

The function is called with the optimal prices, as `participation(-5, 50)`. Since `pA = -5`, the `- pA` term in the buyer equation becomes +5.

Round	nA computation	nA	nB computation	nB
start	—	0	—	0
1	20 + 0.25 · 0 + 5	25	80 + 0.75 · 25 − 50	48.8
2	20 + 0.25 · 48.8 + 5	37.2	80 + 0.75 · 37.2 − 50	57.9
3	20 + 0.25 · 57.9 + 5	39.5	80 + 0.75 · 39.5 − 50	59.6
4	20 + 0.25 · 59.6 + 5	39.9	80 + 0.75 · 39.9 − 50	59.9
5–8	…	40	…	60

Table 6.6: Trace of `participation(-5, 50)` (in thousands, rounded to one decimal).

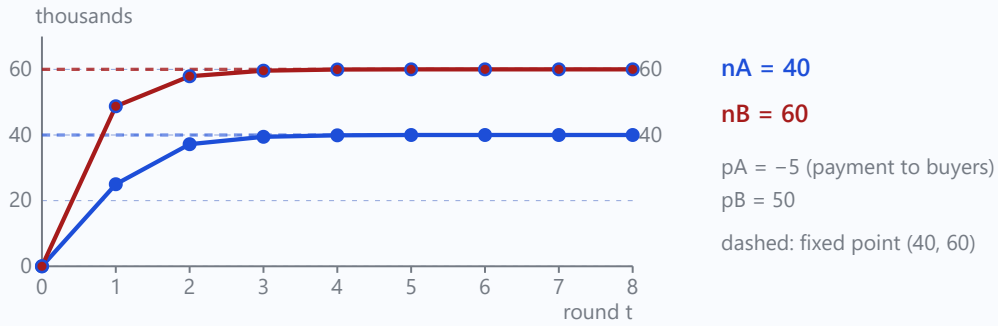


Figure 6.7: Participation by round during `participation(-5, 50)`. Buyers are blue (n_A), sellers red (n_B). Both curves settle within a few rounds on the dashed fixed-point lines at 40 and 60.

Table 6.6 and Figure 6.7 show two properties of the process. First: in the first round the buyer subsidy draws in sellers. 25 thousand buyers bring 48.75 thousand sellers to the platform even though the fee is 50 TL. Second: the cross effect grows the two sides together. In each round the increase in buyers increases sellers, and the increase in sellers increases buyers. The process arrives at the fixed point (40, 60). This point is the **participation equilibrium**, where the expectations of the two sides confirm each other. The profit is $-5 \cdot 40 + 50 \cdot 60 = 2800$.

Why does it converge? The deviation from the fixed point shrinks by a fixed factor every round. The distance of the number of sellers from 60 in the first round is 11.25 (60 − 48.75). This distance is multiplied by $e_A = 0.25$ when passing through the buyer equation and by $e_B = 0.75$ when passing through the seller equation. In one round the deviation falls by a factor of $e_A \cdot e_B = 0.25 \cdot 0.75 = 0.1875$: 11.25 → 2.11 → 0.40 → 0.07. After five rounds the difference is invisible at this rounding.

The convergence condition is the product of the cross effects: $e_A \cdot e_B < 1$. If the product exceeded 1, the deviation would grow each round, and participation would explode or collapse. This is the case in which the network effect is too strong and the tipping behavior of Section 6.2.4 appears. The fixed point itself still exists but is now unstable, like the threshold 0.2 in the adoption model.

Common mistake

Using the old `nA` value in the second line of the trace. The code runs in order: when `nB` is computed in round 1, `nA` is already 25. The correct value is $80 + 0.75 \cdot 25 - 50 = 48.75$. The answer 30 (= 80 − 50) forgets the contribution of the buyers, and the answer 18.75 forgets the constant term and the fee. It should also be remembered that the prices do not change during the loop: convergence is guaranteed not by a price update but by the condition $e_A \cdot e_B < 1$.

6.3.7 The model in the game

The Two-Sided Platform Pricing game uses the same model with two additional parameters:

$$n_A = base_A - s_A \cdot p_A + e_A \cdot n_B$$
$$n_B = base_B - s_B \cdot p_B + e_B \cdot n_A$$

- **base**: the number of users who join when the fee is 0 and the other side is empty.
- **s**: the price sensitivity, that is, how many users a 1-unit increase in the fee loses. In the marketplace scenario $s_A = 11$ and $s_B = 5$; buyers are much more price-sensitive. In the numerical model $s = 1$ was used.
- Participation is limited to 0–100 users; fees are integers between −5 and 10.
- The profit is $\Pi = p_A \cdot n_A + p_B \cdot n_B$, and participation is again the fixed point of mutual reaction (the game's code computes it with a loop like `participation()`).

Level	e_A / e_B	Most profitable (p_A, p_B)	n_A / n_B	Π
1	0.10 / 0.30	(2, 4)	25.3 / 22.6	140.8
2	0.15 / 0.60	(0, 6)	50.3 / 35.2	211.0
3	0.20 / 0.85	(−2, 9)	78.3 / 56.6	352.5

Table 6.7: The most profitable integer fees at each level in the game's marketplace scenario (base 45 / 35).

Table 6.7 shows the counterpart of the rule in the game. The more value side B places on A (as e_B grows), the lower A's most profitable fee becomes: 2, then 0, then −2. At level 3, A is subsidized. At this level, charging both sides the same fee, for example (3, 3), brings in a profit of only 167.0. Subsidizing A and charging B a high fee more than doubles the profit. In the equilibrium at level 3, the amount paid to A, $2 \cdot 78.3 \approx 157$, is small next to the amount collected from B, $9 \cdot 56.6 \approx 509$.

Price sensitivity modifies the rule slightly. Carrying out the same derivation with the coefficients s gives $p_A = n_A/s_A - e_B \cdot n_B/s_B$. The markup reverts to n_A/s_A: the markup of a price-sensitive side is small. This is the source of the statement in the principle that "the more price-sensitive side is charged less." The second term of the rule is also divided by B's sensitivity: if side B is not very price-sensitive, the additional fee that can be collected from it is large. If the fees at level 3 did not have to be integers, the most profitable fees would be about *(−2.1, 9.5)*; these values also obey the rule.

6.3.8 Common mistakes in pricing

Common mistake

"The crowded side is charged less." Crowding is not the criterion for a subsidy. In the numerical example sellers (60 thousand) outnumber buyers (40 thousand), but it is the buyers who are subsidized. What matters is the value a participant adds to the other side.

"A subsidy is a loss." A subsidy is an investment. Looking at one side alone, the platform loses 200 on buyers; but these buyers make it possible for sellers to pay 50 TL. Profit must be computed as a whole.

Confusing the markup with the network effect. In the rule, n_A is A's markup and $e_B\, n_B$ is the value A adds to B. The term e_A (the value B adds to A) does not appear in the formula for p_A; it is in the formula for p_B. The subscript indicates the side that receives the value.

6.4 Sponsored search

Search ads are among the largest sources of revenue for platforms and are sold through an auction held billions of times a day. When several slots are sold at once, the choice of payment rule matters: GSP and VCG bring in different revenue for the same bids, and in GSP truthful bidding is not an equilibrium. In real systems, quality scores and budget constraints also come into play.

6.4.1 The sponsored search model

For every search query, an auction is held among the advertisers who have bid on that keyword. The items sold are ad slots. The slots are not identical: the top slot is clicked more. The auction sells not a single item but several ranked items at once.

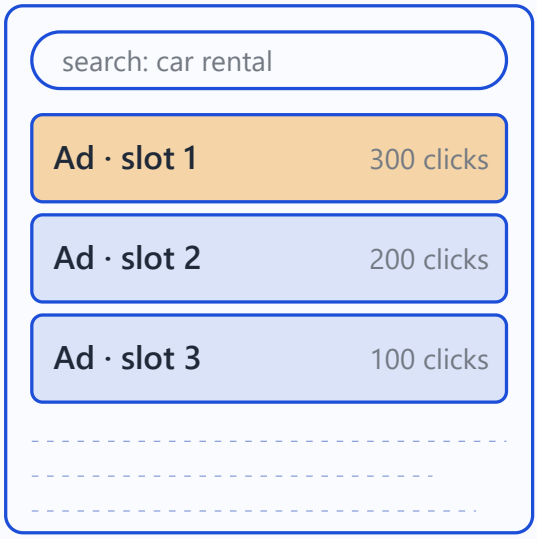


Figure 6.8: Three ad slots for the search "car rental." Per thousand impressions, the slots receive 300, 200, and 100 clicks respectively.

Definition 6.7

Sponsored search auction. There are m slots, and the click counts from top to bottom are $\alpha_1 > \alpha_2 > \ldots > \alpha_m$. Advertiser i has a value per click of v_i. The advertiser bids b_i per click; if placed in slot j they pay p_j per click. The utility of the advertiser in slot j is

$$\alpha_j \cdot (v_i - p_j)$$

An advertiser who gets no slot has a utility of 0.

The utility formula has two factors. α_j is the number of clicks the advertiser receives. $v_i - p_j$ is the profit left over from each click: the value of the click minus the payment per click. A higher slot brings more clicks but is usually more expensive. The advertiser's question is "more clicks, or cheaper clicks?" This tension is the source of strategic behavior in GSP.

All the numerical examples in this section use the same data: three slots, with 300, 200, and 100 clicks per thousand impressions; four advertisers, with values per click of 10 TL for A, 8 for B, 5 for C, and 2 for D. Initially everyone is assumed to bid their value: $b = v$.

6.4.2 Two payment rules: GSP and VCG

Both rules rank the ads by bid; the highest bid goes to the top slot. The difference is only in the payment. In the notation, $b_{(k)}$ denotes the kth highest bid: $b_{(1)} \geq b_{(2)} \geq \ldots$.

Definition 6.8

Generalized Second Price (GSP) auction. The ads are ranked by bid. The advertiser in slot j pays the bid of the one below them per click:

$$p_j = b_{(j+1)}$$

GSP looks like the natural generalization of the single-item second-price auction. With a single slot ($m = 1$) it is indeed identical to the Vickrey auction: the winner pays the second-highest bid. With several slots, however, truthfulness is lost. The reason is shown in Section 6.4.5.

Definition 6.9

VCG auction. The ranking is the same. The advertiser pays the loss their presence imposes on the others. The total payment of the advertiser in slot j is

$$\alpha_j \cdot p_j = \Sigma_{k \geq j} (\alpha_k - \alpha_{k+1}) \cdot b_{(k+1)}$$

where $\alpha_{m+1} = 0$. Here $\alpha_j \cdot p_j$ is the total payment and p_j is the payment per click. In VCG, truthful bidding is a dominant strategy.

The VCG rule is an application of a general principle. A participant's payment is the cost that their presence imposes on the other participants:

$$\textit{payment} = \textit{(others' welfare without them)} - \textit{(others' welfare with them)}$$

This cost is called the **externality**. A participant who pays the externality maximizes their own utility and total welfare at the same time. Misreporting their value cannot gain them anything. The truthfulness of VCG comes from this alignment.

The sum in the formula is the form this principle takes in sponsored search. What would happen without the advertiser in slot j? Those above would be unaffected and would stay where they are, because they already bid higher. Each of those below would move up one slot. The advertiser in slot $k + 1$ would move up to slot k and gain $\alpha_k - \alpha_{k+1}$ extra clicks. Each click is worth $b_{(k+1)}$ to them (equal to their value if they bid truthfully). The total is the sum of these losses from $k = j$ to the end. That is, in VCG the advertiser in slot j pays the sum of the click values that those below them lose by being pushed down one slot each.

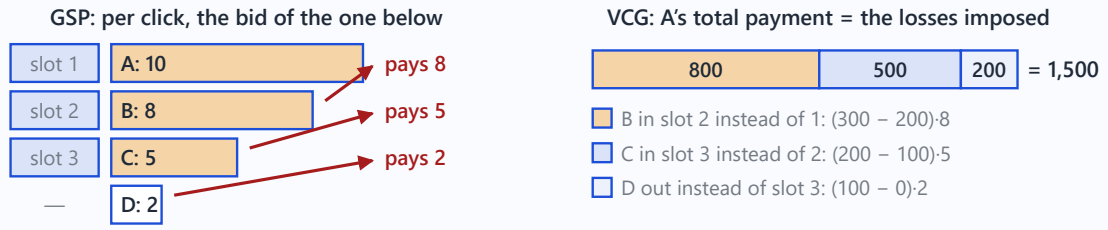


Figure 6.9: Left: the GSP payment ladder: each advertiser pays the bid of the one below per click (8, 5, 2). Right: the components of A's VCG payment: B's loss from staying in slot 2 instead of slot 1 is $(300 - 200) \cdot 8 = 800$, C's is $(200 - 100) \cdot 5 = 500$, and D's is $(100 - 0) \cdot 2 = 200$; the total is 1,500.

Figure 6.9 shows the two rules side by side. In the ladder on the left, each arrow goes from the bid below to the payment of the slot above. The bar on the right shows that A's total VCG payment consists of three parts. Each part is the click value that an advertiser below A loses because of A.

6.4.3 GSP and VCG payments for the same bids

With bids equal to values (bid = value), the payments under the two rules are computed step by step. The ranking is by bid: A goes to slot 1, B to slot 2, and C to slot 3; D is left out.

Example 6.8

GSP. Each slot pays the bid of the one below per click.

1. Slot 1: A pays the bid of B below: *8* TL per click × 300 clicks = 2400.
2. Slot 2: B pays C's bid: $5 \times 200 = 1000$.
3. Slot 3: C pays D's bid: $2 \times 100 = 200$. GSP revenue is $2400 + 1000 + 200 = 3600$.

VCG. Payment = others' welfare "without them" − others' welfare "with them." Starting from the bottom makes the computation easier.

1. **C (slot 3).** Without C, D would move up to slot 3: the others' welfare (only those below) is $100 \cdot 2 = 200$. With C present, D is out: 0. C pays $200 - 0 = 200$, which is $200/100 = 2$ per click.
2. **B (slot 2).** Without B, C would move up to slot 2 ($200 \cdot 5 = 1000$) and D to slot 3 ($100 \cdot 2 = 200$): a total of 1200. With B present, C is in slot 3: $100 \cdot 5 = 500$, and D is out. B pays $1200 - 500 = 700$, which is 3.5 per click.
3. **A (slot 1).** Without A, B, C, and D would each move up a slot: $300 \cdot 8 + 200 \cdot 5 + 100 \cdot 2 = 2400 + 1000 + 200 = 3600$. With A present, it is $200 \cdot 8 + 100 \cdot 5 = 1600 + 500 = 2100$. A pays $3600 - 2100 = 1500$, which is 5 per click.
4. **Check with the formula.** $(300 - 200) \cdot 8 + (200 - 100) \cdot 5 + (100 - 0) \cdot 2 = 800 + 500 + 200 = 1500$ ✓. VCG revenue is $1500 + 700 + 200 = 2400$.

> The welfare of the advertisers above never changes in any of these computations. Only those below are counted, because an advertiser's departure affects only those below them.

Slot	Clicks	Advertiser	GSP /click	GSP total	VCG total	VCG /click
1	300	A (b = 10)	8	2400	1500	5
2	200	B (b = 8)	5	1000	700	3.5
3	100	C (b = 5)	2	200	200	2
—	0	D (b = 2)	—	0	0	—
		Revenue		3600	2400	

Table 6.8: GSP and VCG payments for the same bids (per thousand impressions, in TL).

Table 6.8 yields two observations. First: for the same bids, GSP charges at least as much as VCG in every slot: *8 ≥ 5*, *5 ≥ 3.5*, *2 ≥ 2* TL per click. This is always the case. VCG's payment per click is a weighted average of the bids below, and this average cannot exceed the bid just below. In the bottom slot the two payments are always equal. Second: revenue is 3600 in GSP and 2400 in VCG. But this comparison is not fair. In GSP, truthful bidding is not in the advertiser's interest (Section 6.4.5). Revenues should be compared at each rule's own equilibria (Section 6.4.7).

6.4.4 Code trace: gsp_vcg()

Both payment rules are computed by a single short function. The function takes the list of bids (`bids`) and the slot click counts (`ctr`), and returns the total GSP and VCG payments for each slot.

```
def gsp_vcg(bids, ctr):
  m = len(ctr)
  b = sorted(bids, reverse=True) + [0]*m
  a = ctr + [0]     # dummy slot: 0 clicks
  gsp, vcg = [0]*m, [0]*(m + 1)
  for j in range(m):
    gsp[j] = a[j] * b[j+1]   # next bid
  for j in range(m-1, -1, -1):  # bottom up
    extra = (a[j] - a[j+1]) * b[j+1]
    vcg[j] = vcg[j+1] + extra
  return gsp, vcg[:m]
```

The logic of the code, line by line, is as follows:

- `m = len(ctr)` : the number of slots. 3 in the example.
- `b = sorted(bids, reverse=True) + [0]*m` : the bids are sorted from largest to smallest, and `m` zeros are appended. This way a "next bid" exists for every slot; even if there are fewer advertisers than slots, `b[j+1]` does not run past the end of the list.

- `a = ctr + [0]` : a dummy *(m+1)*th slot with 0 clicks is added. This is the assumption $\alpha_{m+1} = 0$ in the formula.
- `gsp, vcg = [0]*m, [0]*(m + 1)` : the result lists are zeroed. The extra last element (0) of `vcg` is the starting value of the recursion.
- The first loop: `gsp[j] = a[j] * b[j+1]` . The click count of slot *j* times the bid below.
- The second loop goes from the bottom up: `extra = (a[j] - a[j+1]) * b[j+1]` is the click value lost by the advertiser one slot below, who stays one slot lower instead of taking this one. `vcg[j] = vcg[j+1] + extra` : the payment of slot *j* is the payment of the slot below plus this extra loss. The losses imposed on those below are thus carried upward.
- `return gsp, vcg[:m]` : the results are returned with the dummy slot's zero dropped.

The second loop does not recompute the sum in the VCG formula from scratch for each slot. The total for slot *j* is found by adding a single term to the total for slot *j + 1*. Both loops therefore finish in *O(m)* steps; together with sorting, the total work is *O(n log n)* (where *n* is the number of advertisers).

When the function is called as `gsp_vcg([10, 8, 5, 2], [300, 200, 100])` , the trace is as follows:

Loop	j	Computation	Result
GSP	0	a[0] · b[1] = 300 · 8	gsp[0] = 2400
GSP	1	a[1] · b[2] = 200 · 5	gsp[1] = 1000
GSP	2	a[2] · b[3] = 100 · 2	gsp[2] = 200
VCG	2	ek = (100 − 0) · 2 = 200	vcg[2] = 0 + 200 = 200
VCG	1	ek = (200 − 100) · 5 = 500	vcg[1] = 200 + 500 = 700
VCG	0	ek = (300 − 200) · 8 = 800	vcg[0] = 700 + 800 = 1500

Table 6.9: Trace of `gsp_vcg([10, 8, 5, 2], [300, 200, 100])` . b = [10, 8, 5, 2, 0, 0, 0], a = [300, 200, 100, 0].

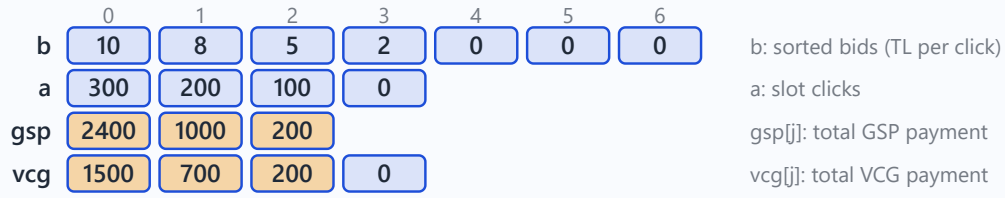


Figure 6.10: The final state of the lists after `gsp_vcg` has run. `b` : the sorted bids and the appended zeros; `a` : the click counts and the dummy 0; `gsp` and `vcg` : the total payment of each slot.

The result is seen in Figure 6.10: GSP [2400, 1000, 200] (total 3600), VCG [1500, 700, 200] (total 2400). This matches the columns of Table 6.8. In a single-slot call (`gsp_vcg([10, 8, 5], [100])`) both lists are [800]: with one slot, GSP and VCG coincide.

6.4.5 Truthful bidding is not an equilibrium of GSP

For a bid profile to be a Nash equilibrium, no advertiser may be able to raise their utility by changing their bid while the others are held fixed. The truthful profile fails this test. To see this, it is enough to look at A's options.

Example 6.9

1. **Utilities in the truthful profile.** Everyone bids their value. A is in slot 1 and pays B's bid (8): $300 \cdot (10 - 8) = 600$. B is in slot 2 and pays 5: $200 \cdot (8 - 5) = 600$. C is in slot 3 and pays 2: $100 \cdot (5 - 2) = 300$. D is out: 0. Platform revenue is 3600.
2. **A's utility as a function of the bid.** With the others (8, 5, 2) fixed, A's bid only determines which slot A lands in. Below 2, A is left out: 0. Between 2 and 5, A lands in slot 3 and pays D's 2: $100 \cdot (10 - 2) = 800$. Between 5 and 8, A lands in slot 2 and pays C's 5: $200 \cdot (10 - 5) = 1000$. Above 8, A stays in slot 1 and pays 8: 600.
3. **Best response.** The largest utility is 1000, obtained with a bid between 5 and 8. For example, if A bids 6, A drops to slot 2 and pays C's 5: $200 \cdot (10 - 5) = 1000 > 600$.
4. **Conclusion.** Truthful bidding is not a best response for A. The truthful profile is not a Nash equilibrium.

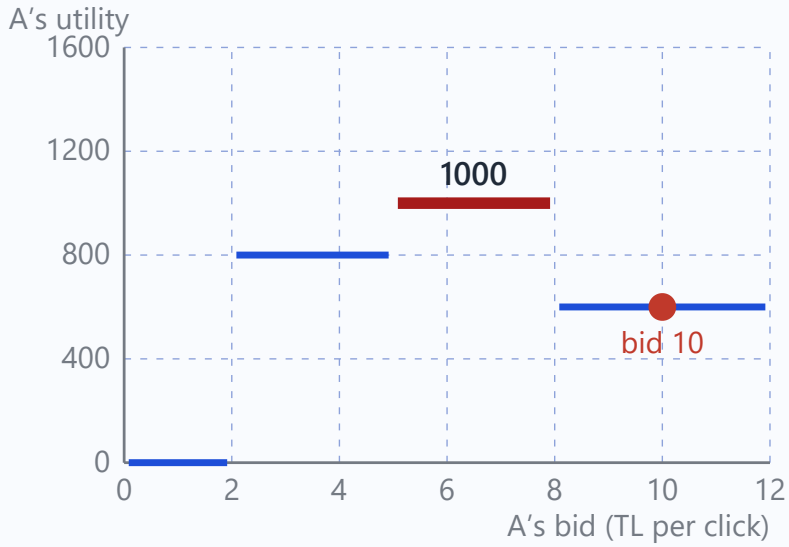


Figure 6.11: With the others truthful (8, 5, 2), A's utility as a function of the bid is a step function: slot 3 between 2 and 5 (800), slot 2 between 5 and 8 (1000), slot 1 above 8 (600). The red dot is the truthful bid of 10; the highest step (red) lies below it.

Figure 6.11 plots A's utility function. The function is a step function because A's payment depends not on A's own bid but on the bid below. When the bid changes within a step, neither the slot nor the payment changes. The steps change only when the bid passes one of the other bids (2, 5, 8). The truthful bid falls on the rightmost step, but this step is not the highest.

Why does this happen? Slot 1 brings only 100 more clicks than slot 2 (300 versus 200). For these extra clicks, A's payment per click rises from 5 to 8. Moreover, this higher price is paid not only for the extra 100 clicks but for *all* 300 clicks. The marginal cost of moving up exceeds the marginal benefit. A gets fewer clicks but pays less per click: $1000 > 600$.

This observation also explains why the single-item Vickrey auction is truthful. With a single item, the loser's utility is 0; there is no "fewer but cheaper" option. With several slots, every lower slot is such an option. In GSP an advertiser can lower their bid and so change the payment rule itself (which bid they will be below). In VCG, the payment equals the loss imposed on the others, whichever slot the advertiser is in. Lowering the bid only changes the slot, and this change never raises total welfare, and therefore never raises A's utility.

An equilibrium example. If the truthful profile is not an equilibrium, what does an equilibrium of GSP look like? One candidate is this: A bids 10, B 5, C 3.5, D 2. The ranking does not change (A, B, C) but the payments fall.

Adv.	v	bid	slot	pays/click	utility
A	10	10	1	5	1500
B	8	5	2	3.5	900
C	5	3.5	3	2	300
D	2	2	—	—	0
Platform revenue					2400

Table 6.10: A Nash equilibrium in GSP. Bids: A 10, B 5, C 3.5, D 2.

Example 6.10

That the profile in Table 6.10 is a Nash equilibrium is shown by trying every deviation of every advertiser.

1. **A (utility 1500).** Any bid above 5 gives slot 1 and a payment of 5: 1500. Between 3.5 and 5, slot 2: $200 \cdot (10 - 3.5) = 1300$. Between 2 and 3.5, slot 3: $100 \cdot (10 - 2) = 800$. Below 2: 0. None exceeds 1500.
2. **B (utility 900).** To get slot 1, B would have to bid more than 10 and pay 10: $300 \cdot (8 - 10) = -600$. For slot 3, B bids between 2 and 3.5 and pays 2: $100 \cdot (8 - 2) = 600$. Out: 0. B does not deviate.
3. **C (utility 300).** For slot 2, C bids between 5 and 10 and pays B's 5: $200 \cdot (5 - 5) = 0$. For slot 1, C would have to pay more than 10: negative. Out: 0. C does not deviate.
4. **D (utility 0).** For slot 3, D would have to bid above 3.5 and pay 3.5: $100 \cdot (2 - 3.5) < 0$. D does not deviate.

No deviation pays; the profile is a Nash equilibrium. The script in the presentation performs the same check for all four advertisers on a grid of bids with step 0.25 and finds no profitable deviation.

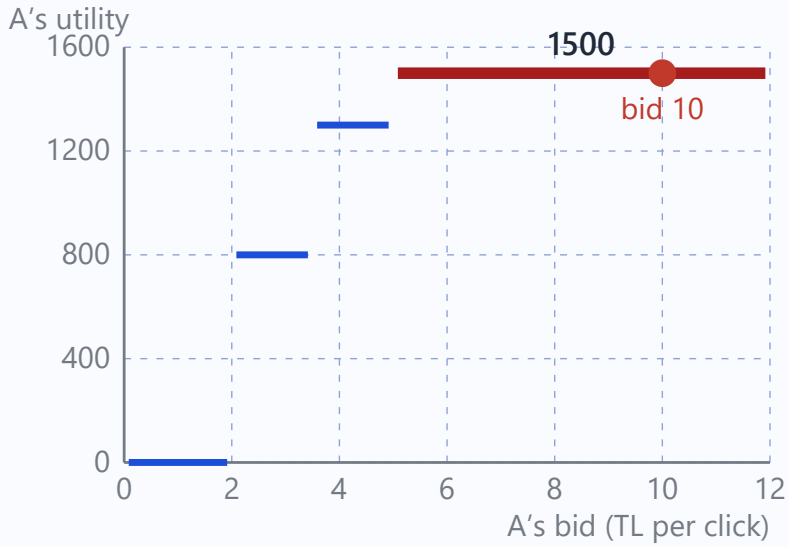


Figure 6.12: At the equilibrium (B 5, C 3.5, D 2), A's utility as a function of the bid: 800 between 2 and 3.5, 1300 between 3.5 and 5, 1500 above 5. A's bid of 10 is on the highest step.

Figure 6.12 shows A's situation at the equilibrium. The steps have changed because the others' bids have changed. Now the highest step is the rightmost, and A's bid is there. In this equilibrium the payments per click are 5 / 3.5 / 2. These are exactly the VCG payments in Table 6.8, and the revenue is 2400. This is no coincidence; the reason is explained in Section 6.4.7.

6.4.6 Code trace: profitable_dev()

Checking whether a profile is an equilibrium means searching for a profitable deviation for each advertiser. The function below performs this search for one advertiser.

```
def profitable_dev(i, b, v, ctr, cands):
  old = gsp_utility(i, b, v, ctr)
  for x in cands:
    b2 = b[:]; b2[i] = x   # only i changes
    u = gsp_utility(i, b2, v, ctr)
    if u > old:
      return x, u   # profitable
  return None     # no deviation
```

The function uses a helper `gsp_utility(i, b, v, ctr)`. This helper sorts the bids; if i is in slot j it returns `ctr[j] · (v[i] − the bid below)`, and if i is out of the slots it returns 0. The bids are assumed to be distinct; how to rank equal bids requires a separate tie-breaking rule.

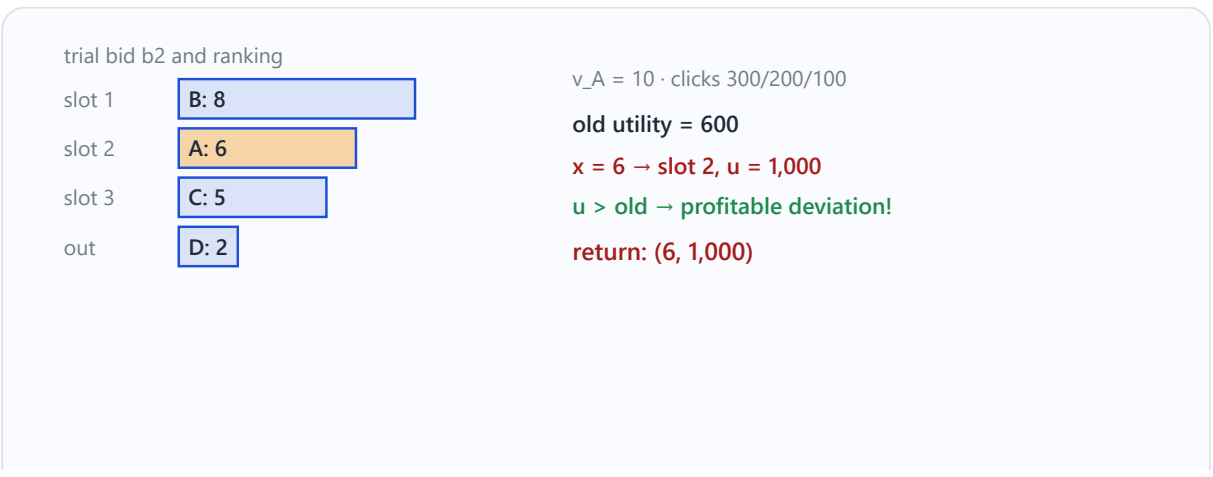
`profitable_dev` works as follows:

- `old`: i's utility with the current bids.
- For each candidate bid `x`, the bid list is copied (`b2 = b[:]`) and only i's bid is changed. Copying matters: if the original list were modified, later candidates would be tested against the wrong profile.
- `u` is i's utility in the new profile. If `u > old`, a profitable deviation has been found, and `(x, u)` is returned.
- If no candidate is profitable, `None` is returned.

The function stops at the first profitable deviation; it finds *a* profitable deviation, not the best one. For an equilibrium check this is enough, because a single profitable deviation suffices to show that the profile is not an equilibrium. By contrast, a result of `None` is valid only for the candidates tried. To say that a profile is an equilibrium, the candidates must cover all the meaningful bids. With a step utility function, one candidate from each step suffices: the candidates in the example (9, 6, 4, 3, 1) place at least one bid on each step of the truthful profile and of the equilibrium.

Profile b	old	x	A's slot	u	Result
[10, 8, 5, 2]	300 · (10 − 8) = 600	9	1	600	600 ≤ 600, continue
		6	2	1000	1000 > 600 → `return (6, 1000)`
[10, 5, 3.5, 2]	300 · (10 − 5) = 1500	9	1	1500	continue
		6	1	1500	continue
		4	2	1300	continue
		3	3	800	continue
		1	out	0	`return None`

Table 6.11: Trace of `profitable_dev(0, b, v, ctr, [9, 6, 4, 3, 1])` for A in two profiles (v = [10, 8, 5, 2], ctr = [300, 200, 100]).



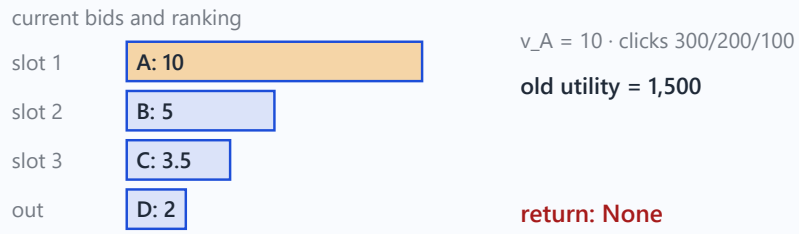


Figure 6.13: Two moments of the `profitable_dev` trace. Top: in the truthful profile, A's bid of 6 drops A to slot 2 and $u = 1{,}000 > 600$; the function returns `(6, 1,000)`. Bottom: in the equilibrium profile no candidate exceeds 1,500; the function returns `None`.

Table 6.11 and Figure 6.13 show the difference between the two calls. In the truthful profile the first candidate (9) does not change the utility, because A is still in slot 1 and still pays 8. The second candidate (6) drops A to slot 2 and the function returns at once. In the equilibrium profile all five candidates are tried and none exceeds 1500. If the same test is run for all four advertisers (`None` for B, C, and D as well), the profile is a Nash equilibrium.

The cost of this function is small: one sort per candidate, that is, $O(n \log n)$. For K candidates and n advertisers, a full check requires $O(K \cdot n^2 \log n)$ operations. In a real search engine the advertiser has a similar computation at hand: bid management tools find the most profitable bid with a search of this kind, given the competitors' bids.

6.4.7 Given bids and equilibrium bids

GSP results are always computed with given bids. 3600 is the revenue in the profile in which everyone bids truthfully. But truthful bidding is not a dominant strategy in GSP, so 3600 is not a real prediction. Putting three profiles side by side makes the situation clear.

Profile	Bids (A, B, C, D)	A's slot and utility	Platform revenue
Truthful	10, 8, 5, 2	slot 1: 300 · (10 − 8) = 600	3600
A bids lower	6, 8, 5, 2	slot 2: 200 · (10 − 5) = 1000	3000
Locally envy-free equilibrium	10, 5, 3.5, 2	slot 1: 300 · (10 − 5) = 1500	2400 = VCG

Table 6.12: Three bid profiles for the same values (A 10, B 8, C 5, D 2).

In the second row, B moves up to slot 1 and pays A's bid (6): $300 \cdot 6 = 1800$. A in slot 2 pays 5: 1000. C in slot 3 pays 2: 200. Revenue is 3000. A gets fewer clicks in the lower slot but pays less per click: $1000 > 600$.

The third row is the equilibrium of Table 6.10. GSP generally has many Nash equilibria. Edelman, Ostrovsky, and Schwarz (2007) singled out a reasonable subclass among these equilibria.

Definition 6.10

If a GSP bid profile is a Nash equilibrium and no advertiser prefers taking the slot above at that slot's current price per click to their own slot, the profile is called a **locally envy-free equilibrium**. For the advertiser in slot j the condition is:

$$\alpha_j \cdot (v - p_j) \geq \alpha_{j-1} \cdot (v - p_{j-1})$$

The condition must hold for every slot (and, for the first advertiser without a slot, against the lowest slot). A profile in which no advertiser envies any slot is called **envy-free**.

The meaning of the condition is this. Nash equilibrium looks only at deviations that can be made alone. For B to move up to slot 1, B would have to exceed A's bid (10) and pay 10; this is unprofitable. But in B's eyes the "current price" of slot 1 is 5 (what A pays). Envy-freeness requires that nobody even think, "I would like to be in A's place at the price A pays." This is a stronger condition, and it describes equilibria that survive in repeated bidding environments: if someone envies the one above, they can raise their own bid and so increase the payment of the one above; this pressure breaks the equilibrium.

Example 6.11

1. **B (slot 2).** In place, B gets $200 \cdot (8 - 3.5) = 900$. If B took slot 1 at its current price of 5, B would get $300 \cdot (8 - 5) = 900$. There is no gain; B does not envy.
2. **C (slot 3).** In place, $100 \cdot (5 - 2) = 300$. If C took slot 2 at 3.5, $200 \cdot (5 - 3.5) = 300$. C does not envy.
3. **D (out).** In place, 0. If D took slot 3 at 2, $100 \cdot (2 - 2) = 0$. D does not envy.

Every advertiser is exactly indifferent toward the slot above. The envy-freeness condition holds with equality. This is the distinguishing feature of the lowest-revenue envy-free equilibrium: if the prices were a penny lower, someone would envy the one above.

Theorem 6.2 (Edelman–Ostrovsky–Schwarz, Varian)

In the model of this chapter (complete information, slot click-through rates independent of the advertiser, constant values per click, and more advertisers than slots), the GSP auction always has a locally envy-free equilibrium. In every locally envy-free equilibrium, GSP revenue is at least VCG revenue. In the one of these equilibria with the lowest revenue, the slots are allocated as in VCG and every advertiser pays the VCG price per click.

The proof of the theorem is outside the scope of these notes (Edelman, Ostrovsky, and Schwarz, 2007; Varian, 2007). But how the lowest equilibrium is constructed can be read off the example: the bid of the advertiser in slot $j + 1$ is set equal to the VCG price per click of slot j. B's bid is the VCG price of slot 1, which is 5; C's is the VCG price of slot 2, which is 3.5; D's is the VCG price of slot 3, which is 2. Since in GSP everyone pays the bid of the one below, the payments equal the VCG payments.

As a result, in these equilibria GSP revenue is at least 2400, and the lowest equals VCG. The 3600 found with truthful bids is not the revenue at an equilibrium. GSP's advantage of "more revenue with the same bids" largely disappears once the bids change.

6.4.8 Comparing GSP and VCG

	GSP	VCG
Is truthful bidding dominant?	No (A bids 6 instead of 10)	Yes
Revenue with the same bids	3600 (≥ VCG in every slot)	2400
Revenue at equilibrium	≥ 2400 at locally envy-free equilibria; the lowest equals VCG	2400 (with truthful bids)
Explanation	Easy: "the bid of the one below"	A sum of externalities; harder
In practice	Common in search ads for a long time (with quality scores)	Used in Facebook's ad auction (Varian and Harris, 2014)

Table 6.13: Comparison of GSP and VCG (with the numbers from the example).

Table 6.13 summarizes the pros and cons of the two rules. VCG is cleaner in theory: the advertiser need not think strategically; reporting their value is enough. GSP, on the other hand, is easy to explain and was historically established first. When a platform changes its rule, the habits and bidding software of thousands of advertisers have to change too; a rule change is therefore costly. In practice the ranking also uses the ad's click quality (the quality score) along with the bid (Section 6.4.10).

The general lesson of this comparison is this: the name "second price" does not guarantee truthfulness. A property that holds for a single item can break with several slots. Whether a mechanism is truthful is learned not from its name but from the best-response computation.

6.4.9 Budgets and online allocation

Real advertisers have daily budgets. Moreover, searches arrive one at a time, and the platform must immediately give each one to an advertiser without knowing which searches will come in the future. This kind of decision is called an **online** decision: the input arrives piece by piece, and each piece is processed irrevocably the moment it arrives.

Example 6.12

A and B each have a budget of 100 TL. For "running shoes" searches both bid 1 TL; for "sports bag" searches only A bids 1 TL. First 100 shoe searches arrive, then 100 bag searches. The platform gives each search to an advertiser whose budget suffices and earns the amount of the bid.

1. **The greedy rule** (highest bid, ties to A). In the shoe searches the two bids are equal; all 100 go to A. A's budget runs out. In the bag searches only A had bid, but no money is left: 0. Revenue is 100.
2. **The Balance rule**: the search goes to the advertiser with the larger remaining budget. The shoe searches are split alternately between A and B: A 50, B 50. In the bag searches A's remaining 50 TL is used: 50. Revenue is *100 + 50 = 150*.
3. **The optimum with knowledge of the future.** If the bag searches had been known in advance, all 100 shoe searches would go to B and all 100 bag searches to A. Revenue is 200.

Rule	Shoe searches	Bag searches	Revenue
Greedy (highest bid, ties to A)	all 100 to A; A's budget runs out	A has no money: 0	100
Balance: to the one with the larger remaining budget	A 50, B 50	A's remaining 50	150
Optimum with knowledge of the future	all 100 to B	all 100 to A	200

Table 6.14: Three rules in online allocation with budgets.

Table 6.14 shows the weakness of the greedy rule. Choosing the highest bid in each search is right for each decision taken alone. But when budgets exist, the greedy rule exhausts the flexible advertiser (A, who bids on several searches) early. Then, when the searches that only A values arrive, nobody can take them.

The success of online algorithms is measured by the **competitive ratio**: the ratio of the algorithm's revenue to the optimum with knowledge of the future, on the worst search sequence. The greedy rule guarantees only 1/2 of the optimum in the worst case; this example is exactly that worst case (100/200). When bids are small relative to budgets, the MSVV rule, which weights each bid by the fraction of the budget remaining, obtains at least $1 - 1/e \approx 63\%$ of the optimum whatever the order of searches (Mehta, Saberi, Vazirani, and Vazirani, 2007). In the example the Balance rule obtained 75%; the guarantee is 63%, but on some sequences better is achieved. Here $e \approx 2.718$ is the base of the natural logarithm.

Common mistake

"Choosing the highest bid on every search is the best." This is true if there are no budgets. With budgets, giving one search to someone consumes money that this advertiser could have used for future searches. In online allocation every decision also affects the future; rules that take the remaining budget into account therefore give better guarantees.

6.4.10 GSP with quality scores

Up to here, every ad was assumed to receive the same number of clicks in a slot. In reality some ads are more engaging than others. Because a platform's revenue comes from clicks, putting an ad at the top that bids high but that nobody clicks is bad for the platform. The quality score solves this problem.

Definition 6.11

GSP with quality scores: the **quality** q_i of ad i is its click propensity. In slot j, the number of clicks for i is $\alpha_j \cdot q_i$. The platform ranks ads by the **score** $b_i \cdot q_i$. The advertiser in slot j pays per click the lowest bid needed to keep their position:

$$p_j = score_{j+1} / q_i = b_{(j+1)}\, q_{(j+1)} / q_i$$

The meaning of the score is simple: $b \cdot q$ measures the expected revenue an impression will bring the platform. The bid is the payment per click, and quality is proportional to the click probability; their product is the expected payment per impression. The payment formula comes from the same logic. To keep its position, advertiser i needs its score to be no less than the score of the one below: $b_i\, q_i \geq score_{j+1}$. This is equivalent to $b_i \geq score_{j+1}/q_i$. GSP's principle of "pay the lowest bid needed to keep your position" turns the boundary of this inequality into the payment. If all qualities are 1, the formula reduces to ordinary GSP.

Example 6.13

The same four advertisers and the same bids (A 10, B 8, C 5, D 2). The qualities are A 0.4, B 0.8, C 0.6, D 1. The slot click counts are $\alpha = 300, 200, 100$.

1. **Scores.** A: $10 \cdot 0.4 = 4$; B: $8 \cdot 0.8 = 6.4$; C: $5 \cdot 0.6 = 3$; D: $2 \cdot 1 = 2$.
2. **Ranking.** B (6.4) > A (4) > C (3) > D (2). A, the highest bidder, is no longer first.
3. **Payments.** B in slot 1, the score below is 4: $4/0.8 = 5$ TL per click. A in slot 2, the score below is 3: $3/0.4 = 7.5$ TL per click. C in slot 3, the score below is 2: $2/0.6 \approx 3.33$ TL per click.

4. **Never above the bid.** Every payment is below the bid: $5 \le 8$, $7.5 \le 10$, $3.33 \le 5$. This is always so, because the advertiser's score is greater than the score of the one below.
5. **Clicks.** B: $300 \cdot 0.8 = 240$; A: $200 \cdot 0.4 = 80$; C: $100 \cdot 0.6 = 60$. The total is 380.
6. **Revenue.** $240 \cdot 5 + 80 \cdot 7.5 + 60 \cdot 3.33 = 1200 + 600 + 200 = 2000$.
7. **Comparison: ranking by bid alone.** The order becomes A, B, C. The clicks are $300 \cdot 0.4 = 120$, $200 \cdot 0.8 = 160$, $100 \cdot 0.6 = 60$; the total is 340. The payments under ordinary GSP are 8, 5, 2: revenue is $120 \cdot 8 + 160 \cdot 5 + 60 \cdot 2 = 960 + 800 + 120 = 1880$.



Figure 6.14: GSP with quality scores. The bars are the $b \cdot q$ scores; the order is B (6.4), A (4), C (3), D (2). On the right, the payments per click: B $4/0.8 = 5$, A $3/0.4 = 7.5$, C $2/0.6 = 3.33$; all below the bid.

Ranking	slots 1–3	total clicks	revenue (TL)
By bid alone	A, B, C	340	1,880
With quality score (b·q)	B, A, C	380	2,000

Table 6.15: Effect of the quality score on clicks and revenue (per thousand impressions).

Table 6.15 shows two effects of the quality score. In this example the quality score increased both clicks (340 → 380) and revenue (1,880 → 2,000). Users see the relevant ad, and the platform sells more clicks. In addition, the high-quality ad reaches the same slot more cheaply: B pays 5 per click in slot 1, whereas under ranking by bid alone, A in slot 1 paid 8. Quality is rewarded; the advertiser is thus encouraged to improve the ad.

The quality score also answers the second of the opening questions of the game: the highest bidder does not always get the top slot. A is in the second slot even though A bid 10.

Common mistake

Taking the payment in GSP with quality scores to be $b_{(j+1)}$ (the bid below). The correct payment is the *score* of the one below divided by the advertiser's own quality. In the example, B pays not the bid of A below (10) but $4/0.8 = 5$. If the bid below were paid, B would pay more than its own bid (8).

6.4.11 Back to the game

The first question at the opening was this: if a click earns the advertiser 8 TL, is bidding 8 TL the best choice? The theory's answer is no. With the click-through rates of the game's "Search Ads" scenario (12%, 7%, 3.5%), the slots get 120, 70, and 35 clicks per thousand impressions. Let the advertiser's value per click be 8, the rivals' bids 6, 4, and 2, and all qualities 1. At Level 2 of the game the rule is GSP.

Bid	Slot	Payment/click	Utility
8 (truthful)	1	6	120 × 2 = 240
5	2	4	70 × 4 = 280
3	3	2	35 × 6 = 210

Table 6.16: Results of three bids in the game when the value is 8 (rivals 6, 4, 2; GSP).

Table 6.16 repeats the logic of A's example. The truthful bid gets slot 1 but pays 6 per click; 2 is left from each click. A bid of 5 gets slot 2; it pays 4 per click and 4 is left from each click. Although the number of clicks falls from 120 to 70, total utility rises from 240 to 280. Dropping to a lower slot is more profitable. The game's "Find best response" button searches for exactly this deviation; that is, it does what the `profitable_dev` function does.

The same behavior has been observed in the real world. In the first-price auction used in the early days of sponsored search (Level 1 of the game: everyone pays their own bid), bids kept rising and then collapsing; advertisers would bid just above each other, and then, when this became unprofitable, would fall back (Edelman and Ostrovsky, 2007). The GSP used by Google reduced this fluctuation, but in GSP too truthful bidding is not dominant (Edelman, Ostrovsky, and Schwarz, 2007). Platforms can also raise revenue by setting a reserve price (the lowest accepted bid); Ostrovsky and Schwarz (2011) showed this with a field experiment.

The summary of the ad part of the chapter is this: the name "second price" brings truthfulness with a single slot; with several slots it does not. The rule that guarantees truthfulness is VCG.

6.4.12 Common mistakes in ad auctions

Common mistake

Mistaking GSP for Vickrey. "GSP is a second-price auction, so truthful bidding is dominant." This is true only with a single slot. With several slots, truthful bidding may sometimes be a best response, but not always.

Counting those above in a VCG payment. An advertiser's departure affects only those below. In A's VCG payment the losses of B, C, and D are counted; in C's payment only D is counted.

Confusing total and per-click payment. The VCG formula gives $\alpha_j \cdot p_j$, that is, the total payment. For the payment per click, the total is divided by the slot's click count: for B, *700/200 = 3.5*.

Comparing revenues under the wrong profile. "GSP brings in 50% more revenue than VCG (3600 versus 2400)." This comparison gives the two rules the same bids. In GSP the bids change; at the lowest envy-free equilibrium the two rules have the same revenue.

6.5 Reputation and competition

Platforms bring strangers together. A buyer buys a product from a seller they have never met; a rider gets into the car of a driver they have never seen. Where does this trust come from? Largely from reputation systems. Reputation systems reduce information asymmetry, but they must be designed carefully to rank items with few reviews fairly and to make fake reviews unprofitable. Which platform users stay on is determined by competition, regulation, and recommendation systems.

6.5.1 Reputation systems and information asymmetry

Definition 6.12

If in a transaction one party holds important information the other does not know, there is **information asymmetry**. The typical situation in an online market is this: the seller knows the quality of the product, and the buyer does not know it in advance.

Akerlof (1970) showed with the example of used cars how information asymmetry can collapse a market. When quality is unknown, the buyer offers an average price. This price does not satisfy the good seller, who withdraws from the market. Only bad products remain on the market. Since defective cars are called "lemons" in American slang, this situation is called the **market for lemons**.

Example 6.14

Suppose half the cars on the market are good and half are bad. A good car is worth 8 to its seller and 10 to a buyer (in thousands of TL). A bad car is worth 3 to its seller and 4 to a buyer. For every car the buyer's value is higher than the seller's; that is, every car changing hands benefits both sides.

1. **If quality were known.** A good car would sell at a price between 8 and 10, and a bad car at a price between 3 and 4. All cars would be sold.
2. **If quality is unknown.** The buyer cannot tell whether a car is good or bad. With a fifty-fifty probability the expected value is $0.5 \cdot 10 + 0.5 \cdot 4 = 7$. The buyer offers at most 7.

3. **The good seller withdraws.** 7 is below 8, the good seller's value. The good seller does not sell; only bad sellers accept 7.
4. **The buyer anticipates this.** A buyer who knows that only bad cars remain on the market offers at most 4. In the end only bad cars change hands; the gain of 2 on each good car is lost.

Example 6.14 applies to online markets too. If the buyer cannot know that a seller is good, they will not pay a price worth the good seller's effort, and the good seller leaves the platform. A reputation system exists to fill this information gap, and it does two jobs:

- **Selection:** finding the good seller. The experiences of earlier buyers give a new buyer information about the seller's quality.
- **Discipline:** punishing bad behavior in the future. A seller who ships a poor-quality product today loses customers tomorrow because of a low rating. This is the logic of cooperation in repeated games: today's cheating is paid for with the loss of future earnings.

Reputation systems also give rise to their own strategic problems:

Fake reviews
: Purchased positive reviews or negative reviews written about a rival. Luca and Zervas (2016) showed, on the restaurant review site Yelp, that fake reviews are related to competitive conditions: businesses whose reputation is weakening and that face intense competition resort to fake reviews more.

Fear of retaliation
: In mutual rating (the host rates the guest and the guest rates the host), bad experiences go unreported, because whoever reports fears getting a bad rating in return.

Rating inflation
: If everyone gets 4.8 and above, ratings are not discriminating. A rating of 4.6 means "below average."

Selective participation
: Those who are moderately satisfied mostly stay silent; reviews cluster at the extremes. A new seller, meanwhile, may receive no reviews at all; without reviews they cannot stand out, and because they cannot stand out they receive no sales or reviews. This is called the **cold start** problem.

There are design measures against these problems: a verified-purchase requirement (only someone who actually bought the product can write a review), giving less weight to old reviews, accounting for the uncertainty of a rating with few reviews (the Bayesian average), and, in mutual rating, revealing both sides' ratings at the same time. The last measure removes the fear of retaliation: nobody gives their rating after seeing the other's, so a rating cannot be a tool of retaliation.

6.5.2 The Bayesian average

In ranking products, one often meets this question: is a product with a 5.0 rating from 2 reviews better, or one with a 4.7 rating from 200 reviews? By raw average the first is ahead. But two reviews are very little information; two happy customers, or two fake reviews, can make the average 5.0.

Definition 6.13

Bayesian average: the score of a product with n reviews is computed as

$$(C \cdot m + \Sigma \text{ ratings}) / (C + n)$$

Here m is the platform's overall average (3.5 in the example), C is a confidence constant (10 in the example), and Σ *ratings* is the sum of the ratings the product received.

The formula is easy to interpret: each product is treated as if C "average" virtual reviews (with rating m) had been added to it. The virtual reviews pull the average of a product with few reviews toward the platform average. A product with many reviews, on the other hand, stays almost at its own average, because 10 virtual reviews are small next to hundreds of real ones.

The same formula can also be written as a weighted average:

$$\text{Bayesian score} = w \cdot (\text{raw average}) + (1 - w) \cdot m,\ w = n / (n + C)$$

The weight w shows how much the product's own data is trusted. With 2 reviews, $w = 2/12 \approx 0.17$: only 17% of the score comes from the product's own average. With 200 reviews, $w = 200/210 \approx 0.95$. The name "Bayesian" comes from this: the platform average is used as a **prior** (prior belief), and the product's reviews are the evidence that updates this belief. As the evidence grows, the posterior approaches the evidence.

Example 6.15

Three products: X (2 reviews, average 5.0), Y (200 reviews, average 4.7), Z (20 reviews, average 4.2). $m = 3.5$, $C = 10$, so $C \cdot m = 35$.

1. **Raw ranking.** X (5.0) > Y (4.7) > Z (4.2). X has only 2 reviews.
2. **Rating sums.** X: $2 \cdot 5 = 10$; Y: $200 \cdot 4.7 = 940$; Z: $20 \cdot 4.2 = 84$.
3. **Bayesian scores.** X: $(35 + 10)/(10 + 2) = 45/12 = 3.75$. Y: $(35 + 940)/(10 + 200) = 975/210 \approx 4.64$. Z: $(35 + 84)/(10 + 20) = 119/30 \approx 3.97$.
4. **New ranking.** Y (4.64) > Z (3.97) > X (3.75). X, with few reviews, is pulled toward the average and falls to last.

5. **A fake review attempt.** Suppose 3 fake 5★ reviews are added to X: 5 reviews, total 25. The raw average is still 5.0; X stays at the top of the raw ranking. The Bayesian score is *(35 + 25)/(10 + 5) = 60/15 = 4.0*; X is still below Y.

Product	reviews	average	rank	Bayesian score	rank
X	2	5	1	3.75	3
Y	200	4.7	2	4.64	1
Z	20	4.2	3	3.97	2
X + 3 fake	5	5	1	4	2

Table 6.17: Raw-average and Bayesian-average rankings (m = 3.5, C = 10).

Table 6.17 shows the main benefit of the Bayesian average: it reduces the effect of fake reviews. In the raw average, X needs no fake reviews at all to reach the top; two real or fake reviews are enough. In the Bayesian score, for X to catch up with Y we need $(45 + 5k)/(12 + k) \geq 975/210$, which gives $k \geq 30$. Many more fake reviews are needed to create the same effect.

The choice of constants is a design decision. As C grows, products with few reviews are penalized more; resistance to fraud increases, but genuinely good new products are also noticed late. m is usually the average of the platform or of the category. Choosing a low m makes a platform cautious toward new products, and a high one makes it optimistic.

Common mistake

Writing the average in place of the sum in the Bayesian formula. The numerator is $C \cdot m + \Sigma$ *ratings*, that is, the *sum* of the product's ratings is added. For Y the numerator is not $35 + 4.7$ but $35 + 940$. If the average is given, first multiply by the number of reviews to get the sum.

6.5.3 When are fake reviews profitable?

Although fake reviews are prohibited, some sellers resort to them. From the mechanism design viewpoint the question is: under what conditions are fake reviews profitable for a seller? Fake reviewing can be modeled as an investment decision: it has a return, a cost, and an expected penalty.

Example 6.16

A seller has 20 reviews and an average of 3.6. Sellers with a rating of 4.0 or above are shown near the top of search results; a top position brings the seller an extra profit of $G = 600$ TL per month. A fake 5★ review costs 15 TL each. The platform detects fraud with probability ρ; a seller who is caught loses $L = 3000$ TL (the account is suspended and sales are lost).

1. **How many fake reviews are needed (raw average)?** The rating sum is $20 \cdot 3.6 = 72$. After adding k fake 5 ★ reviews the average is $(72 + 5k)/(20 + k)$. The condition is $(72 + 5k)/(20 + k) \geq 4$. Multiplying by the denominator gives $72 + 5k \geq 80 + 4k$, that is, $k \geq 8$. Check: for $k = 8$, $112/28 = 4.0$ ✓; for $k = 7$, $107/27 \approx 3.96$, which is not enough.
2. **Cost.** $8 \times 15 = 120$ TL.
3. **Expected gain.** The seller expects an extra profit of 600 TL, pays 120 TL, and with probability ρ loses 3000 TL: $600 - 120 - \rho \cdot 3000$.
4. **Profitability condition.** Cheating is profitable if the expected gain is positive: $480 > \rho \cdot 3000 \iff \rho < 0.16$. For example, at $\rho = 0.1$ the expected gain is $480 - 300 = 180$ TL; at $\rho = 0.2$ it is $480 - 600 = -120$ TL.
5. **With the Bayesian score.** With $m = 3.5$ and $C = 10$, the numerator is $35 + 72 = 107$ and the denominator is 30; the seller's current Bayesian score is $107/30 \approx 3.57$. The condition is $(107 + 5k)/(30 + k) \geq 4$: $107 + 5k \geq 120 + 4k$, that is, $k \geq 13$. The cost is $13 \times 15 = 195$ TL. Profitability: $600 - 195 > \rho \cdot 3000 \iff \rho < 0.135$.

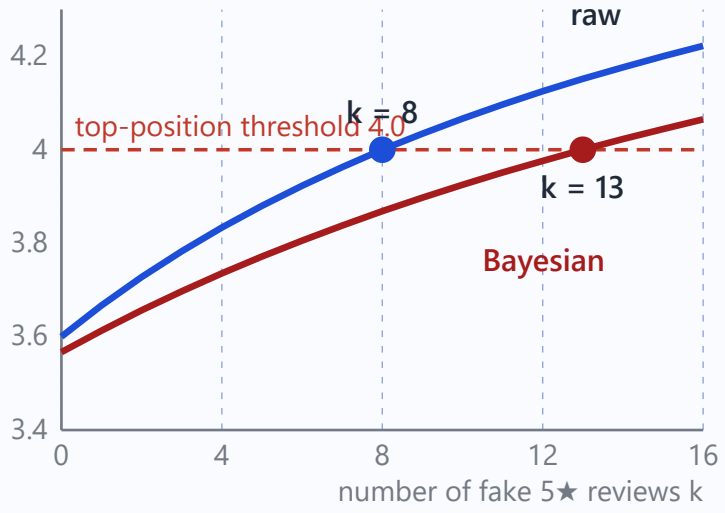


Figure 6.15: Raw average (blue) and Bayesian score (red) as a function of the number of fake 5★ reviews k. The dashed line is the top-position threshold of 4.0. The raw average reaches the threshold at $k = 8$, and the Bayesian score at $k = 13$.

Figure 6.15 compares the two curves. The Bayesian curve starts below the raw curve (3.57 versus 3.6) and rises more slowly, because each fake review has to outweigh the 10 virtual reviews along with the real ones. The same threshold therefore requires 13 fake reviews; cheating becomes more expensive and the profitability boundary falls to $\rho < 0.135$.

The model clearly shows the three levers in the platform's hands. The expected gain has the form G – *(cost of reviews)* – $\rho \cdot L$, and the platform can influence every term:

- **Raise ρ:** a verified-purchase requirement, anomaly detection (many 5★ reviews arriving on the same day, reviews from new accounts).
- **Raise L:** closing the account, wiping the seller's accumulated reputation, fines.
- **Lower G or raise the cost of reviews:** robust scoring (the Bayesian average) reduces the effect of fake reviews; more fake reviews are needed for the same ranking gain.

The mechanism design view is clear here: the platform does not content itself with merely banning cheating; it pushes the expected return of cheating below zero. A rational seller looks not at the rule itself but at the rule's effect on their own gain.

Common mistake

Not rounding a fractional k up. If the inequality gives $k \geq 7.5$, 7 fake reviews are not enough; since the number must be an integer, 8 are needed.

Forgetting the probability of punishment. "A fake review brings 600 TL for 120 TL, so it is always profitable." The expected penalty $\rho \cdot L$ must be taken into account; when $\rho \geq 0.16$, cheating brings a loss.

6.5.4 Platform competition and regulation

Network effects determine not only the growth of a single platform but also competition between platforms. When the tipping dynamics of Section 6.2 are applied to a market with two platforms, the result is this: users tend to go to whichever is more crowded, and the more crowded one becomes more crowded still. This shapes three features of competition.

Winner takes all
: Because of tipping, a single platform can take most of the market. Even if the second platform's product is good, its value is low if its users are elsewhere.

Switching costs
: Friends, past ratings, accumulated data, and habits keep users in place. A seller with hundreds of positive reviews does not want to start from scratch on a new platform. Switching costs let a platform raise its fees without losing its users.

Multi-homing
: Users can use several platforms at once: a driver can work for both Uber and a rival. Multi-homing weakens monopolization. Since the small platform can reach the same drivers, its value to riders does not fall; tipping weakens.

Examples from Türkiye and around the world show this dynamic: Hepsiburada and Trendyol, Uber and Lyft, Instagram and TikTok. In these pairs both platforms have managed to survive. One reason is multi-homing: sellers sell on two marketplaces at once, and users use two apps at once. Another reason is local network effects: a platform can be strong in some cities or among some groups of users.

The concentration created by network effects also raises ethical and regulatory questions:

- **Data privacy and security:** large platforms collect the data of a great many people.
- **The risk of monopolization and self-preferencing:** if a marketplace is also a seller, it can promote its own products in the ranking. The platform becomes both referee and player.
- **Transparency of ranking and recommendation algorithms:** a seller or content creator cannot know why they are invisible.

The European Union's **Digital Markets Act (DMA)** is a regulation aimed at these problems. The act defines platforms that have a very large user base and that stand in a key position between business users and consumers as **gatekeepers**, and imposes obligations on them. For example, a gatekeeper may not rank its own products above those of its rivals and must allow users to take their data to other platforms. From a game-theoretic viewpoint, the regulation is a constraint imposed from outside on the rules the platform sets for itself: a higher-level mechanism that the mechanism designer must also obey.

Data portability and interoperability weaken tipping by lowering switching costs. In terms of the network effect model, this means that users can take the value they have accumulated on one platform to a new platform; the new platform's initial expectation x_0 then does not start from zero.

6.5.5 Recommendation, loyalty, and privacy

A platform's recommendation and ranking algorithm allocates visibility: it decides who stands out. This is an **allocation rule**, just as an ad auction allocates slots. Participants choose their strategies according to this rule. Four topics stand out.

The platform's objective
: If the algorithm maximizes short-term engagement or revenue, it can conflict with the user's long-term benefit. Clickbait headlines and infinite feeds raise engagement metrics but consume the user's time and attention. The platform's objective function and the user's utility are not the same thing.

Creators are players too
: Creators and sellers compete for visibility and gravitate toward the format the algorithm rewards (title, tags, video length). When everyone chases the same reward, content can become uniform. This is the collective outcome of participants best-responding to the rule.

Loyalty programs
: Loyalty points, paid memberships (for example fast shipping), and gamification (badges, streaks) deliberately raise switching costs and reduce multi-homing. A user who has paid for a membership tends to concentrate their shopping on that platform, because the value of the membership grows only as they shop there.

Data and privacy
: The price of a "free" service is data. More data allows better recommendations but also raises the risk of leaks and misuse. The user also chooses strategically how much data to share: they trade off better recommendations against less privacy.

The common question of these four topics is the question of the whole chapter: are the platform's goals aligned with the interests of users and creators? If they are not, users following their own interests may produce outcomes that neither the platform nor society wants. The task of mechanism design is to establish this alignment.

6.6 Discussion

The concepts of the chapter answer three general questions. Each question rests on a model computed in these notes.

6.6.1 Why do some platforms grow fast while others collapse?

Critical mass and tipping answer this question. If the expectation is above the threshold, the feedback loop grows the platform; if below, it collapses it. In the adoption model, the difference of 60 users between two starting points turned into the difference between 800 users and zero. Ease of use, trust, and early incentives make it easier to cross the threshold: ease of use and incentives lower p, while trust and promotion raise x_0. The same mechanism works in reverse: when a platform that is losing users falls below the threshold, the collapse accelerates.

6.6.2 How do platforms shape behavior?

Bonuses and discounts increase particular behaviors: extra pay for drivers in a busy area draws drivers to that area. Recommendation algorithms steer preferences: what is visible is chosen. Spam and fake-review controls limit manipulation: the probability of detection and the penalty lower the expected return of cheating. In every case the user best-responds to the platform's rule; behavior is a function of the rule.

6.6.3 Can a design be fair to everyone?

Perfect fairness is hard. The first reason is that users' preferences differ: a price structure that subsidizes buyers costs sellers more. The second reason is that network effects favor large groups: a user in a large city gets better service than one in a small city, because network effects are local. The third reason is strategic manipulation: even a fair rule can be abused. Still, the situation can be improved with incentives, rules, and regulation. The Bayesian average both protects and penalizes products with few reviews; the quality score rewards the ad that gets clicked; the DMA limits a platform's favoring of its own products.

Common mistake

Seeing platform problems only as technical faults or ill intent. Fake reviews, bonus abuse, and content uniformity are often the result of rational behavior, that is, of the best response to the

rule. The solution is usually not to blame the users but to change the rule.

6.7 Summary

- An **online platform** is an intermediary that brings the sides together; its value comes from the sides' access to one another (a two-sided market). The platform's rules (who joins, who sees whom, who pays what) form a mechanism; user behavior is the Nash equilibrium under these rules.
- A **network effect** can be direct, cross-side, negative, or local. A positive network effect sets up a feedback loop, and the market can tip in one direction.
- In the **adoption model** ($\theta \sim U[0, 1]$, utility $\theta x - p$) the fraction that joins is $F(x) = 1 - p/x$. The equilibria are the solutions of $F(x) = x$: for $p = 0.16$, 0 and 0.8 are stable and 0.2 is unstable. The unstable equilibrium is the **critical mass**: if the expectation is above 0.2 the market goes to 0.8, and if below, to 0. If $p > 0.25$, the market never forms.
- The **price structure** is how the fee is split between the sides. In the linear model the profit-maximizing price is $p_A = n_A - e_B\, n_B$: the markup minus the value an A participant adds to side B. In the marketplace example $p_A = -5$, $p_B = 50$, $\Pi = 2800$; the buyer is subsidized because the value buyers add for sellers (45) exceeds the markup (40).
- The **participation equilibrium** can be found by letting the sides react to each other in turn (`participation()`); if the product of the cross effects is less than 1 ($0.25 \cdot 0.75 = 0.1875$), the deviation shrinks by this factor each round and the process converges.
- In **GSP** the slot j advertiser pays the bid of the one below per click; truthful bidding is not dominant (A bids 6 instead of 10 and raises its utility from 600 to 1000). In **VCG** the advertiser pays the loss imposed on the others; truthful bidding is dominant. With the same bids, GSP brings in 3600 and VCG 2400.
- At **locally envy-free equilibria** GSP revenue is at least VCG revenue; at the lowest, the payments are the same as in VCG (bids 10, 5, 3.5, 2 → 5 / 3.5 / 2 TL per click).
- The **quality score** ($b \cdot q$) changes the ranking and makes the payment $score_{j+1}/q_i$; in the example it raises clicks from 340 to 380 and revenue from 1,880 to 2,000.
- In **online allocation with budgets** the greedy rule gets half the optimum in the worst case; for budgets with small bids the MSVV rule guarantees $1 - 1/e \approx 63\%$ of it.
- **Reputation systems** reduce information asymmetry (the market for lemons). The **Bayesian average** $(C \cdot m + \Sigma\, ratings)/(C + n)$ pulls a product with few reviews toward the average and makes fake reviews more expensive: the number of fake reviews needed for the top position rises from 8 to 13, and cheating is profitable only for $\rho < 0.135$ instead of $\rho < 0.16$.
- **Competition**: tipping leads to "winner takes all"; switching costs strengthen a monopoly, and multi-homing weakens it. Regulations such as the DMA impose obligations on gatekeeper platforms.

Concept	Description	Example
Network effect, tipping	$F(x) = 1 - p/x$; two stable equilibria and the threshold between them	$p = 0.16$: 0 and 0.8 stable, threshold 0.2
Two-sided pricing	$p_A = n_A - e_B n_B$: the side that adds much value to the other is charged less	$p_A = -5$, $p_B = 50$, $\Pi = 2800$
GSP	Slot j pays the bid of the one below; truthful bidding is not dominant	revenue 3600 with the same bids; A bids 6 instead of 10
VCG	Pays the loss imposed on others; truthful bidding is dominant	1500 / 700 / 200, revenue 2400
GSP equilibrium	The lowest locally envy-free equilibrium gives the VCG outcome	bids 10, 5, 3.5, 2 → 5 / 3.5 / 2 TL per click
Quality score	Rank by $b \cdot q$, pay $score_{j+1}/q_i$	order B, A, C; clicks 380, revenue 2,000
Online allocation	Accounting for the remaining budget beats greedy	greedy 100, Balance 150, optimum 200
Reputation	The Bayesian average makes fake reviews more expensive	fake reviews needed 8 → 13

Table 6.18: The concepts of the chapter at a glance.

6.8 Key terms

online platform two-sided market marketplace network effect direct network effect cross-side (indirect) network effect feedback loop chicken-and-egg problem adoption model self-fulfilling expectation unstable equilibrium critical mass tipping seeding pricing price structure subsidy markup sponsored search auction advertiser slot (ad position) click-through rate (CTR) Generalized Second Price (GSP) auction second-price (Vickrey) auction Vickrey–Clarke–Groves (VCG) mechanism externality dominant strategy profitable deviation envy-free equilibrium (GSP) locally envy-free equilibrium (GSP) quality score online allocation Balance algorithm competitive ratio reputation system information asymmetry market for lemons cold start Bayesian average winner-takes-all switching cost multi-homing self-preferencing Digital Markets Act (DMA) gatekeeper (DMA)

6.9 Test Yourself

1. In the adoption model ($\theta \sim U[0, 1]$, utility $\theta x - p$) the cost of joining is $p = 0.1875$. What are the equilibria, and what is the critical mass? For initial expectations $x_0 = 0.3$ and $x_0 = 0.2$, where do the

first steps of the dynamics lead?

Answer

$x^2 - x + 0.1875 = 0$: the discriminant is $1 - 0.75 = 0.25$, with square root 0.5; the roots are $(1 \pm 0.5)/2$, that is, 0.25 and 0.75. Also $x = 0$. The stable equilibria are 0 and 0.75; the critical mass is 0.25 (250 users). $x_0 = 0.3$ is above the threshold: $x_1 = 1 - 0.1875/0.3 = 0.375$, participation rises and goes to 0.75. $x_0 = 0.2$ is below the threshold: $x_1 = 1 - 0.1875/0.2 = 0.0625$; since this is smaller than p, $x_2 = 0$ and the platform collapses.

2. In a marketplace, $n_A = 30 + 0.25\, n_B - p_A$ and $n_B = 60 + 0.75\, n_A - p_B$ (marginal cost 0). What are the profit-maximizing participation levels, prices, and profit? Is the buyer subsidized? Does the result obey the rule $p_A = n_A - e_B\, n_B$?

Answer

$\Pi = 30\, n_A + 60\, n_B + n_A n_B - n_A^2 - n_B^2$. Partial derivatives: $30 + n_B - 2n_A = 0$, $60 + n_A - 2n_B = 0$. From the first, $n_A = (30 + n_B)/2$; substituting into the second gives $120 + 30 + n_B - 4n_B = 0$, that is, $n_B = 50$, $n_A = 40$. The prices are $p_A = 30 + 12.5 - 40 = 2.5$, $p_B = 60 + 30 - 50 = 40$; the profit is $100 + 2000 = 2100$. The rule: $40 - 0.75 \cdot 50 = 2.5$ ✓, $50 - 0.25 \cdot 40 = 40$ ✓. The buyer is not subsidized: the value buyers add for sellers (37.5) is smaller than the markup (40). The buyer's fee is nonetheless much lower than the seller's.

3. Three slots get 400, 250, and 50 clicks per thousand impressions. Four advertisers bid their value per click: A 9, B 7, C 4, D 3. What are the total payment of each slot and the platform revenue under GSP and VCG? What are the payments per click in VCG?

Answer

GSP: A $400 \cdot 7 = 2800$, B $250 \cdot 4 = 1000$, C $50 \cdot 3 = 150$; revenue 3950. VCG from the bottom up: C $(50 - 0) \cdot 3 = 150$; B $(250 - 50) \cdot 4 + 150 = 950$; A $(400 - 250) \cdot 7 + 950 = 1050 + 950 = 2000$; revenue 3100. Per click: A $2000/400 = 5$, B $950/250 = 3.8$, C $150/50 = 3$. In every slot GSP ≥ VCG (7 ≥ 5, 4 ≥ 3.8, 3 ≥ 3).

4. In the GSP of question 3 everyone bids truthfully. What is A's (value 9) utility? Can A raise its

utility by changing its bid? Is the truthful profile a Nash equilibrium?

Answer

With the truthful bid, A pays 7 in slot 1: $400 \cdot (9 - 7) = 800$. With a bid between 4 and 7, A drops to slot 2 and pays C's 4: $250 \cdot (9 - 4) = 1250 > 800$. Between 3 and 4, A drops to slot 3 and pays 3: $50 \cdot 6 = 300$. The best response is a bid between 4 and 7 (utility 1250). Since A can deviate profitably, the truthful profile is not a Nash equilibrium. The difference in clicks between slots 1 and 2 (150) is small and the difference in price (7 versus 4) is large, so the lower slot looks attractive.

5. Two slots get 100 and 50 clicks per thousand impressions; the number of clicks of i in slot j is $\alpha_j q_i$. Bids and qualities: A 5 (q = 0.6), B 4 (q = 0.9), C 6 (q = 0.4), D 2 (q = 1). In GSP with quality scores, what are the order, the payments per click, the total clicks, and the revenue? What changes compared with GSP that ranks by bid alone?

Answer

Scores: A 3, B 3.6, C 2.4, D 2 → order B, A, C, D. B in slot 1 pays $3/0.9 \approx 3.33$ TL per click, and A in slot 2 pays $2.4/0.6 = 4$ TL per click. Clicks: B $100 \cdot 0.9 = 90$, A $50 \cdot 0.6 = 30$; total 120. Revenue $90 \cdot 10/3 + 30 \cdot 4 = 300 + 120 = 420$. Ranking by bid alone gives C (6), A (5): C gets $100 \cdot 0.4 = 40$ clicks and pays 5 (200), and A gets $50 \cdot 0.6 = 30$ clicks and pays B's 4 (120); total 70 clicks, revenue 320. The quality score raises clicks from 70 to 120 and revenue from 320 to 420; C, the highest bidder, gets no slot.

6. The Bayesian average is computed with $m = 3.5$, $C = 10$. Product P has 5 reviews (average 4.8), and product R has 50 reviews (average 4.4). What are the raw and Bayesian rankings? At least how many fake 5★ reviews must be added to P for its Bayesian score to reach R's?

Answer

Raw ranking: P (4.8) > R (4.4). Bayesian: P $(35 + 24)/(10 + 5) = 59/15 \approx 3.93$; R $(35 + 220)/(10 + 50) = 255/60 = 4.25$. The Bayesian ranking is R > P. With fake reviews: $(59 + 5k)/(15 + k) \geq 4.25 \iff 59 + 5k \geq 63.75 + 4.25k \iff 0.75k \geq 4.75 \iff k \geq 6.33$; since it must be an integer, $k = 7$ ($94/22 \approx 4.27$). 6 fake reviews are not enough ($89/21 \approx 4.24$).

7. A seller has 30 reviews with an average of 3.7. Sellers rated 4.0 or above move to the top position and earn an extra profit of $G = 500$ TL per month. A fake 5★ review costs 10 TL each; the detection probability is ρ, and the loss when caught is $L = 2000$ TL. With the raw average and with the Bayesian score (m = 3.5, C = 10), how many fake reviews are needed, and for which values of ρ is

cheating profitable?

Answer

The rating sum is $30 \cdot 3.7 = 111$. Raw: $(111 + 5k)/(30 + k) \geq 4 \iff 111 + 5k \geq 120 + 4k \iff k \geq 9$. The cost is 90 TL; profitable $\iff 500 - 90 > 2000\rho \iff \rho < 0.205$. Bayesian: $(35 + 111 + 5k)/(40 + k) \geq 4 \iff 146 + 5k \geq 160 + 4k \iff k \geq 14$. The cost is 140 TL; profitable $\iff 360 > 2000\rho \iff \rho < 0.18$. The Bayesian score raises the number of fake reviews needed from 9 to 14 and narrows the range of profitability.

8. A's budget is 80 TL and B's is 40 TL. For "book" searches both bid 1 TL; for "notebook" searches only A bids 1 TL. First 80 book searches arrive, then 80 notebook searches. How much revenue do the greedy rule (ties to A), the Balance rule (to the one with the larger remaining budget; ties to A), and the optimum with knowledge of the future bring in?

Answer

Greedy: all 80 book searches go to A and A's budget runs out; nobody is left for the notebooks: revenue 80. Balance: the book searches go to A until A's remainder falls to 40 (40 searches); after that, since the remainders are equal, the remaining 40 book searches are split 20–20. A has spent 60 and B 20. In the notebook searches A's remaining 20 TL is used: revenue $80 + 20 = 100$. Optimum: 40 of the book searches to B and 40 to A; 40 of the notebook searches to A: revenue 120 (the sum of the two budgets). Greedy gets 2/3 of the optimum and Balance gets 5/6.

6.10 Further reading

- D. Easley and J. Kleinberg, *Networks, Crowds, and Markets* (2010), Chapters 15 and 17: explains sponsored search auctions and the network effect model (critical mass, tipping) with little mathematics and many figures; a draft of the book can be read free on the authors' site.
- T. Roughgarden, Stanford CS364A lecture notes (2013), Lectures 2 and 3: presents the basic concepts of mechanism design and Myerson's lemma through the sponsored search example, briefly and clearly, from a computer science viewpoint.
- B. Edelman, M. Ostrovsky, and M. Schwarz, "Internet Advertising and the Generalized Second-Price Auction" (2007): the foundational paper that introduces the history of GSP, its lack of truthfulness, and locally envy-free equilibria; the introduction and example sections can be read without going into technical detail.
- J.-C. Rochet and J. Tirole, "Platform Competition in Two-Sided Markets" (2003): the paper that introduced the concept of a two-sided market and the importance of the price structure; it discusses the logic of subsidies with examples such as credit cards and software.

6.11 References

1. G. A. Akerlof, "The Market for 'Lemons': Quality Uncertainty and the Market Mechanism", *Quarterly Journal of Economics* 84(3), 488–500, 1970.
2. M. Armstrong, "Competition in Two-Sided Markets", *RAND Journal of Economics* 37(3), 668–691, 2006.
3. D. Easley, J. Kleinberg, *Networks, Crowds, and Markets: Reasoning about a Highly Connected World*, Cambridge University Press, 2010 (Chapters 15 and 17).
4. B. Edelman, M. Ostrovsky, "Strategic Bidder Behavior in Sponsored Search Auctions", *Decision Support Systems* 43(1), 192–198, 2007.
5. B. Edelman, M. Ostrovsky, M. Schwarz, "Internet Advertising and the Generalized Second-Price Auction: Selling Billions of Dollars Worth of Keywords", *American Economic Review* 97(1), 242–259, 2007.
6. M. Luca, G. Zervas, "Fake It Till You Make It: Reputation, Competition, and Yelp Review Fraud", *Management Science* 62(12), 3412–3427, 2016.
7. A. Mehta, A. Saberi, U. Vazirani, V. Vazirani, "AdWords and Generalized Online Matching", *Journal of the ACM* 54(5), 2007.
8. N. Nisan, T. Roughgarden, É. Tardos, V. Vazirani (eds.), *Algorithmic Game Theory*, Cambridge University Press, 2007 (Chapters 27–28).
9. M. Ostrovsky, M. Schwarz, "Reserve Prices in Internet Advertising Auctions: A Field Experiment", *ACM Conference on Electronic Commerce (EC)*, 2011.
10. J.-C. Rochet, J. Tirole, "Platform Competition in Two-Sided Markets", *Journal of the European Economic Association* 1(4), 990–1029, 2003.
11. T. Roughgarden, Stanford CS364A: Algorithmic Game Theory lecture notes, 2013 (Lectures 2–3: sponsored search auctions).
12. H. R. Varian, "Position Auctions", *International Journal of Industrial Organization* 25(6), 1163–1178, 2007.
13. H. R. Varian, C. Harris, "The VCG Auction in Theory and Practice", *American Economic Review: Papers & Proceedings* 104(5), 442–445, 2014.

PART II · CHAPTER 7

Social Choice and Voting

When a group of people must decide together, not everyone wants the same thing. A club elects a president, a committee approves one of several projects, a search engine merges rankings from different sources into a single list. All of these decisions raise the same question: how can a single collective decision be extracted from individual preferences? The field that studies this question is social choice theory. When the most widely used voting rules are applied step by step to a single preference profile, the same ballots produce four different winners. Paradoxes such as the Condorcet cycle, the power of the agenda, and monotonicity violations, the Kemeny rule, and strategic voting, in which voters can change the outcome by lying, are all treated with numerical examples. The theorems of Arrow, Gibbard–Satterthwaite, and May, single-peaked preferences, computational social choice, judgment aggregation, and proportional representation are the remaining topics of these notes.

LEARNING OUTCOMES

Counting
Apply majority, plurality, Borda, approval, instant-runoff, and Condorcet rules step by step to the same preference profile.

Paradoxes
Demonstrate the Condorcet cycle, the effect of the agenda order, and the monotonicity violation of IRV; find the Kemeny ranking.

Strategic voting
Show numerically tactical voting under plurality and the "burying" manipulation under Borda.

Theorems
Explain what the theorems of Arrow, Gibbard–Satterthwaite, and May say; recognize the escape route offered by single-peaked preferences.

7.1 Social choice

In the games of game theory, each player tries to maximize their own payoff, and the outcome results from combining the players' choices. Voting is similar: each voter reports a preference, a rule aggregates these reports, and a single outcome that applies to everyone is produced. The difference is that the outcome is determined by an explicitly written rule. The question of this chapter is: how should this rule be chosen, and what good and bad properties does the chosen rule have?

7.1.1 A game to start with: the Condorcet paradox

The chapter opens with a small game played in class. Three voters rank three candidates, A, B, and C, according to their own tastes. Then every pair of candidates is compared by pairwise majority: when A and B are compared, each voter votes for whichever of the two is higher in their own ranking. A candidate who beats all rivals this way is called a **Condorcet winner**. The name comes from the French mathematician Marquis de Condorcet, who studied such comparisons in the 18th century.

At the first level of the game the voters' rankings are as follows: the first voter has $A \succ B \succ C$, the second $B \succ C \succ A$, and the third $C \succ A \succ B$. Here the symbol "$\succ$" means "is preferred to"; $A \succ B \succ C$ reads "I prefer A to B, and B to C." Figure 7.1 shows the three rankings as columns; the candidate at the top is preferred more. In the triangle on the right, each edge is a pairwise comparison, and the question mark shows that it is not yet known who wins that comparison.

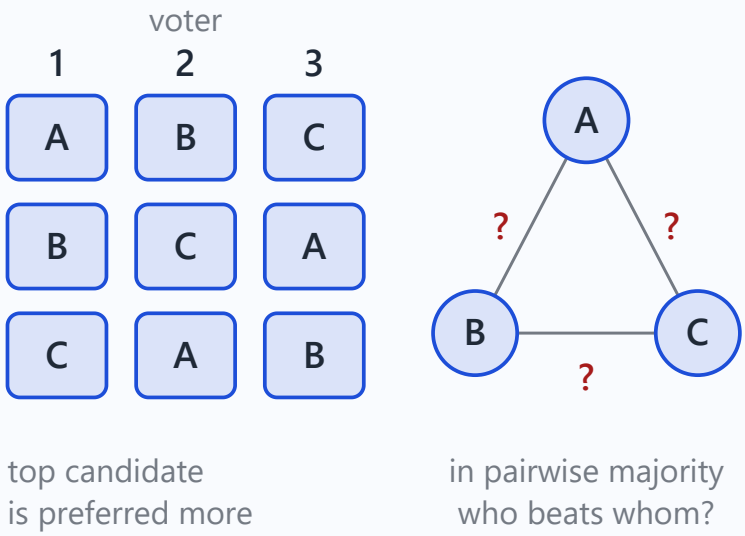


Figure 7.1: The first level of the Condorcet Paradox game. Left: the rankings of the three voters (the candidate at the top is preferred more). Right: the comparison triangle to be filled in with pairwise majority.

The game asks two questions. First: if each voter's ranking is consistent in itself, must the community's preference be consistent too? Second: if one person had the authority to set the order of the votes, could that person make any candidate they like win? The answers to both questions are

computed in Section 7.3. In short: the community's preference can be inconsistent, and the person who sets the voting order can pick the outcome. These two observations show that what is called "the will of the people" is not always automatically well defined.

7.1.2 What is social choice?

Social choice is the field that studies rules that turn individual preferences into a collective decision or a social ranking. The aim of the field is not to find a single universal definition of fairness. The aim is to understand which criteria rules satisfy and what trade-offs they create. A rule may satisfy some good properties while violating others; which property matters more depends on the context of the decision.

The areas of application are broad: general elections, committee and club decisions, choosing which activity a class will go to, deciding how to use a shared resource, and online polls. The same questions arise in computer science. Merging the result lists of several search engines into one, having several classifiers vote on a decision, or producing a ranking from user ratings are all problems of extracting a collective ranking from individual rankings.

The basic object of social choice is the preference. A voter's preference is a ranking that orders the candidates from most liked to least liked. This ranking is assumed to have two properties.

Definition 7.1

Let A be the set of candidates. A voter's **preference** is a ranking over A written $x \succ y$ ("prefers x to y"). This ranking is

- **complete**: for every pair of distinct x, y, either $x \succ y$ or $y \succ x$;
- **transitive**: if $x \succ y$ and $y \succ z$, then $x \succ z$.

Completeness says that the voter can compare any two candidates; there is no "I don't know" answer. Transitivity means that the ranking is free of internal contradiction. If someone who prefers A to B and B to C also prefers C to A, this person cannot give a consistent answer when asked which of A, B, and C to choose: whichever they pick, there is another candidate they like more. In this chapter every voter's preference is assumed to be complete and transitive. One of the chapter's surprising results is this: even if this assumption holds for individuals, it may fail for the community.

Definition 7.2

The list of the preferences of n voters is called a **preference profile**: $(\succ_1, \succ_2, \ldots, \succ_n)$. The ranking a voter reports is called a **ballot**.

The ballot and the true preference are not the same thing. A voter who really thinks A ≻ B ≻ C may write B ≻ A ≻ C on the ballot. From the game-theoretic viewpoint, the ballot is the voter's *strategy*: the voter can choose the ballot that makes the rule's outcome best for their true preference. This viewpoint is the basis of strategic voting in Section 7.4.

A voting rule takes a preference profile as input and produces an outcome. Two kinds of rules are distinguished according to the type of outcome.

Definition 7.3

- A **social choice function (SCF)** assigns a single winning candidate to each profile: $f(\succ_1, \ldots, \succ_n) \in A$.
- A **social welfare function (SWF)** assigns a social ranking of the candidates to each profile: $F(\succ_1, \ldots, \succ_n)$ is a ranking over A.

Figure 7.2 summarizes this structure. On the left is the input, the preference profile; each column is the ranking of a group of voters. In the middle stands the rule f, which assigns an outcome to the profile. On the right are the two kinds of output: a single winner (SCF) at the top and a complete social ranking (SWF) at the bottom. In the example in the figure the rule is the Borda count; the winner is B and the ranking is B ≻ D ≻ A ≻ C. Where these numbers come from is computed step by step in Section 7.2.3.

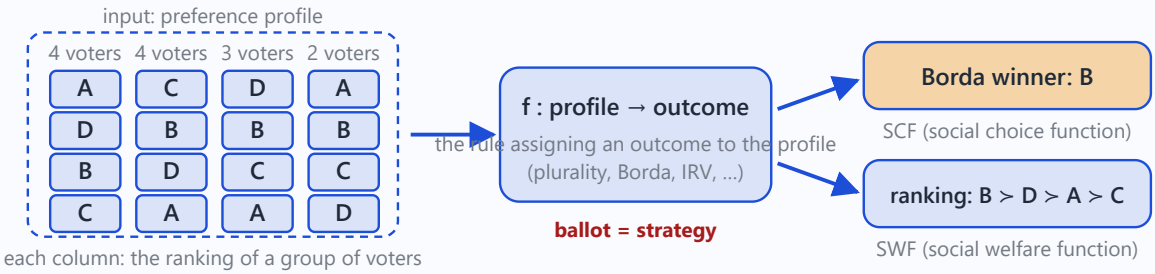


Figure 7.2: The input and output of a social choice rule. The preference profile goes into the rule; the output is either a single winner (SCF) or a social ranking (SWF). The ballot is the voter's strategy.

The note "ballot = strategy" below the figure shows the chapter's link with mechanism design. In mechanism design, a designer asks players to report their private information (their preferences) and picks an outcome according to these reports; often the designer also collects payments from the players or pays them. Voting is the *moneyless* special case of mechanism design: there are no payments, only rankings. Since there is no money, it is impossible to "punish" a voter for lying or to "reward" them for telling the truth. The consequences of this restriction are seen clearly in the Gibbard–Satterthwaite theorem in Section 7.5.4.

Common mistake

Confusing an SCF with an SWF. An SWF always yields an SCF: the first candidate in the social ranking is the winner. The converse is not true: knowing a winner does not give the social order of the remaining candidates. This distinction matters because Arrow's theorem is about SWFs (rankings), while the Gibbard–Satterthwaite theorem is about SCFs (a single winner).

7.1.3 An example preference profile

Most of the rules in this chapter are compared on a single preference profile. This makes it plain that different rules draw different outcomes from the same votes. The profile is given in Table 7.1.

rank	4 voters	4 voters	3 voters	2 voters
1.	A	C	D	A
2.	D	B	B	B
3.	B	D	C	C
4.	C	A	A	D

Table 7.1: An example preference profile: 13 voters, 4 candidates. Each column is a group of voters who gave the same ranking.

The table is read as follows. Voters who gave the same ranking are collected in one column; the column heading says how many voters gave that ranking. The first column means "4 voters gave the ranking $A \succ D \succ B \succ C$." For these four voters, A is the best and C the worst candidate. The four voters in the second column give $C \succ B \succ D \succ A$, the three voters in the third column give $D \succ B \succ C \succ A$, and the two voters in the fourth column give $A \succ B \succ C \succ D$. Throughout the notes these four columns are referred to as *Group 1*, *Group 2*, *Group 3*, and *Group 4* (G1–G4 for short).

The total number of voters is $4 + 4 + 3 + 2 = 13$. In some rules a candidate needs more than half of the votes to win. Half of 13 is 6.5; more than half means at least 7 votes. This number is called the **majority threshold**, and in this profile it is 7.

A few notable features of the profile can be noted in advance. A is the first choice of six voters (G1 and G4) but the last choice of seven voters (G2 and G3): A is a candidate who polarizes the voters. No voter puts B first, but nine voters put B second and nobody puts B last: B is the compromise candidate whom everyone calls "acceptable." D is the first choice of three voters and the second choice of four. These features explain why each rule picks a different candidate: rules applied to the same profile give four different winners.

Example 7.1

Question: how many voters in the profile prefer D to B? For each group, compare the positions of D and B.

1. G1 (A ≻ D ≻ B ≻ C): D is second and B third; D ≻ B. 4 voters.
2. G2 (C ≻ B ≻ D ≻ A): B is second and D third; B ≻ D. 0 voters.
3. G3 (D ≻ B ≻ C ≻ A): D is first and B second; D ≻ B. 3 voters.
4. G4 (A ≻ B ≻ C ≻ D): B is second and D fourth; B ≻ D. 0 voters.

Total: $4 + 3 = 7$ voters prefer D to B, and $4 + 2 = 6$ voters prefer B to D. The two numbers add up to the number of voters (13), because in a complete ranking each voter puts exactly one of the two candidates ahead. This check is the easiest way to catch counting errors in pairwise comparisons.

7.2 Voting rules

All of the most widely used voting rules are applied to the same profile in Table 7.1. Three questions are asked for each rule: what does the voter report, how is the count done, and whom does the rule choose in this profile?

7.2.1 Overview of the basic voting rules

Rules are distinguished by two questions. First: what does the voter report on the ballot? In some rules only a single candidate is written; in some a complete ranking of the candidates is requested, and in others a set of acceptable candidates. Second: how are the reports counted and how is the winner determined? Table 7.2 summarizes six basic rules according to these two questions.

Rule	What does the voter report?	Winner
Majority	A single candidate	The candidate with more than half of the votes (no winner if none)
Plurality	A single candidate	The candidate with the most first choices; need not exceed half
Borda count	Full ranking	For m candidates, rank positions get $m-1, m-2, \ldots, 0$ points; the highest total
Approval	The set of approved candidates	The candidate with the most approvals
Instant-runoff voting (IRV)	Full ranking	The candidate with the fewest first choices is eliminated and their votes are transferred; repeat until a majority
Condorcet	Full ranking	The candidate who beats every rival in pairwise comparison (may not always exist)

Table 7.2: Basic voting rules.

Some of the rules in the table use only a part of the ranking. Majority and plurality look only at the first choice; the voter's second and third choices are not counted at all. Borda, IRV, and Condorcet use the whole ranking. Using more information does not always give a "fairer" outcome, but it changes which information the outcome is sensitive to.

Common mistake

The sentence “the one with the most votes wins” is the definition of plurality, not of majority. Majority rule requires more than 50% of the votes; if this threshold is not exceeded, there is no winner. In everyday language both are called “majority,” but their consequences differ: in the 13-voter profile, a candidate with 6 votes wins under plurality but does not satisfy majority rule.

7.2.2 Majority, plurality, and the two-round system

Definition 7.4

Each voter reports only their first choice. A candidate’s **first-choice count** is the number of voters who put them first.

- **Majority rule**: the candidate whose first-choice count is greater than $n/2$ wins; if there is no such candidate, there is no winner.
- **Plurality rule**: the candidate with the largest first-choice count wins.

Both rules use the same count; the difference is only in how the winner is declared. Majority rule imposes the condition of more than $n/2$ votes. When this condition holds, it is certain that the winner has a majority of the voters behind them. But the condition does not always hold; then the rule cannot produce a winner and an additional method (such as a second round) is needed. Plurality always produces a winner (together with a tie-breaking rule), but the winner may have only a small minority of the voters behind them.

Example 7.2

The first choices are counted for the profile in Table 7.1. Only the first row of the table is looked at: A, C, D, A.

1. A: the 4 voters of G1 and the 2 voters of G4 put A first; $4 + 2 = 6$.
2. B: no group puts B first; 0.
3. C: the 4 voters of G2; 4.
4. D: the 3 voters of G3; 3.
5. Check: $6 + 0 + 4 + 3 = 13$, equal to the number of voters.

Majority: the threshold is 7 and the highest count is $6 < 7$; there is no winner. **Plurality:** A has the most first choices and wins. But A has only 6/13 of the votes behind them, about 46%.

In practice, the most common solution when a majority is not reached is the **two-round system** (runoff). The presidential elections in Türkiye and France use this system. If no candidate achieves a majority in the first round, the two candidates with the most votes in the first round go to the second round. In the second round only these two candidates compete; voters vote for whichever of the two they prefer.

Example 7.3

In the first round A got 6 and C got 4 votes, so A and C go to the second round. Each voter's preference between these two candidates is read from their ranking.

1. G1 (A ≻ D ≻ B ≻ C): A is above C; 4 votes for A.
2. G2 (C ≻ B ≻ D ≻ A): 4 votes for C.
3. G3 (D ≻ B ≻ C ≻ A): their favorite D is not in the race; in their ranking C is third and A fourth, so 3 votes for C.
4. G4 (A ≻ B ≻ C ≻ D): 2 votes for A.

Second round: $A = 4 + 2 = 6$, $C = 4 + 3 = 7$. C wins. A, the leader of the first round, lost the second round, because the 3 voters supporting D prefer C to A.

This example shows why the two-round system can give a different result from plurality: the second round gives the voters of eliminated candidates the opportunity to choose among the remaining candidates. But this opportunity is limited. A candidate who came third or fourth in the first round cannot reach the second round, even if they could beat everyone in a one-on-one race. This is exactly what happens in this profile: as will be seen in Section 7.2.7, D beats all rivals in pairwise comparison, but because D gets only 3 votes in the first round, D cannot reach the second round.

7.2.3 The Borda count

Plurality takes no account of the voter's second choice. A candidate can be everyone's second choice and still receive zero votes. In the 18th century the French sailor and mathematician Jean-Charles de Borda noticed this problem and proposed a count that gives points to every rank.

Definition 7.5

With m candidates, in the **Borda count** each voter reports a full ranking. The candidate in position k in a voter's ranking receives $m - k$ points from that voter: first place $m - 1$, second place $m - 2$, …, last place 0 points. A candidate's Borda score is the sum of the points received from all voters; the candidate with the highest score wins.

The Borda score has a simple interpretation: the points x receives in a voter's ranking equal the number of candidates that voter places below x. The candidate in first place leaves $m - 1$ candidates behind, and the last one leaves none. A candidate's total Borda score is therefore the sum, over all voters, of the number of candidates that x beats. This interpretation helps connect Borda with the pairwise comparisons of Section 7.2.7: the Borda score equals a candidate's row sum in the pairwise comparison table.

For four candidates the points are 3 for first, 2 for second, 1 for third, and 0 for fourth place. A group's contribution to a candidate is "number of voters × points." The count is done group by group.

Example 7.4

1. **G1** (4 voters, A ≻ D ≻ B ≻ C): A $4 \times 3 = 12$, D $4 \times 2 = 8$, B $4 \times 1 = 4$, C $4 \times 0 = 0$.
2. **G2** (4 voters, C ≻ B ≻ D ≻ A): C $4 \times 3 = 12$, B $4 \times 2 = 8$, D $4 \times 1 = 4$, A 0. Running totals: A 12, B 12, C 12, D 12: after two groups the four candidates are tied.
3. **G3** (3 voters, D ≻ B ≻ C ≻ A): D $3 \times 3 = 9$, B $3 \times 2 = 6$, C $3 \times 1 = 3$, A 0. Running totals: A 12, B 18, C 15, D 21.
4. **G4** (2 voters, A ≻ B ≻ C ≻ D): A $2 \times 3 = 6$, B $2 \times 2 = 4$, C $2 \times 1 = 2$, D 0.

The result is collected in Table 7.3: B = 22, D = 21, A = 18, C = 17. The Borda winner is B.

candidate	G1	G2	G3	G4	total
A	12	0	0	6	18
B	4	8	6	4	**22**
C	0	12	3	2	17
D	8	4	9	0	21

Table 7.3: Contribution of the groups and totals in the Borda count (points 3, 2, 1, 0).

The most interesting aspect of the result is that no voter put B first. B's strength is that no voter put B last and that nine voters put B second. Borda promotes such compromise candidates. This can be seen as a virtue (a candidate everyone can accept) or as a flaw (a candidate nobody really wants).

There is an easy way to check the computation. Each voter distributes a total of $3 + 2 + 1 + 0 = 6$ points. 13 voters must distribute a total of $13 \times 6 = 78$ points. Indeed $18 + 22 + 17 + 21 = 78$. If the total does not match, a contribution has been miscomputed somewhere.

Common mistake

- **Starting the points from 1:** using 4, 3, 2, 1 points adds 1 extra point from each voter to every candidate. This does not change the winner, but the results then do not match the numbers in these notes, and it causes confusion when the number of candidates changes

(when a candidate withdraws). The scale in the definition is $m - 1, \ldots, 0$.

- **Forgetting the group size:** a column is not one voter but a group of voters. G3's contribution to D is not 3 but $3 \times 3 = 9$.

7.2.4 Approval voting

Definition 7.6

In **approval voting**, each voter marks the set of candidates they accept (approve); they may approve as many candidates as they like. Each approval gives a candidate 1 point; the candidate with the most approvals wins.

In approval voting, what the voter reports is not a ranking but a set. The voter says, "these candidates are acceptable to me, the others are not." The outcome of approval voting therefore depends not only on the voters' rankings but also on *where each voter draws the line*. The preference profile alone does not determine the approval winner; how many candidates each voter approves must also be known. An assumption is therefore needed for the approval count.

Example 7.5

Assumption: each voter approves the first two candidates in their ranking. The first two rows of the table are read.

1. A: G1 (ranked 1st) 4 and G4 (ranked 1st) 2; $4 + 2 = 6$.
2. B: G2 (ranked 2nd) 4, G3 (ranked 2nd) 3, G4 (ranked 2nd) 2; $4 + 3 + 2 = 9$.
3. C: G2 (ranked 1st) 4; 4.
4. D: G1 (ranked 2nd) 4, G3 (ranked 1st) 3; $4 + 3 = 7$.
5. Check: each voter gives 2 approvals; $13 \times 2 = 26 = 6 + 9 + 4 + 7$.

The approval winner is B (9). If everyone approved only their first choice, the approval counts would be the same as the first-choice counts (A 6, B 0, C 4, D 3) and A would win.

The example shows two important points. First, when everyone approves their top two candidates, approval voting turns into the *2-approval* rule (Section 7.2.5); when everyone approves only their top candidate, it turns into plurality. Approval voting sits flexibly between these two extremes. Second, the result depends on the approval threshold: with the same preferences, when the threshold changes the winner moves from A to B. Since the voter sets the threshold, the approval threshold is also a strategic decision. For example, a voter who supports A, if they know B is a strong rival, may give up approving their second choice B.

Approval voting is used in some scientific society elections and in meeting polls of the "which times are suitable?" kind. In a meeting poll, each participant marks all the times they are available, and the time that suits the most people is chosen; this is exactly approval voting.

7.2.5 Positional scoring rules

Plurality, Borda, and approval voting (when everyone approves the same number of candidates) are in fact different forms of the same idea: a fixed score is given to each rank and the scores are added up. This common structure can be written with a single definition.

Definition 7.7

For m candidates, a **positional scoring rule** is given by a score vector: $s = (s_1, s_2, \ldots, s_m)$, with $s_1 \geq s_2 \geq \ldots \geq s_m$ and $s_1 > s_m$. The candidate in position k of a voter's ranking receives s_k points; the candidate with the highest total score wins.

- **Plurality**: $s = (1, 0, \ldots, 0)$.
- **k-approval**: the first k positions get 1 point, the rest 0; for $k = 2$, **2-approval**.
- **Borda**: $s = (m - 1, m - 2, \ldots, 0)$.
- **Veto rule**: $s = (1, \ldots, 1, 0)$; only the last position gets 0 points.

The scores are in decreasing order so that a voter never gives fewer points to a candidate they like more. The name of the veto rule comes from its interpretation: each voter "vetoes" just one candidate, that is, rejects them; a candidate's veto score is the number of voters who did not put them last. The veto rule looks for the candidate whom everyone objects to least.

Figure 7.3 and Figure 7.4 show the count of the same profile with two different score vectors. On the left is the profile; the orange cells are the positions that receive points. On the right, each candidate's total score is drawn as a bar; the orange bar shows the winner.

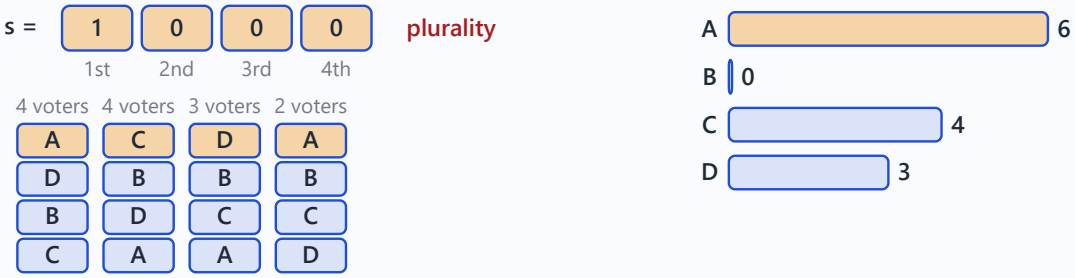


Figure 7.3: Plurality, $s = (1, 0, 0, 0)$. Only first place gets points: A = 6, C = 4, D = 3, B = 0; the winner is A.

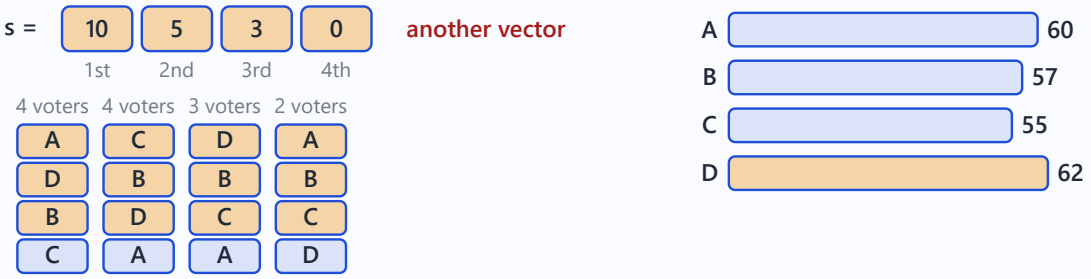


Figure 7.4: Another vector, *s = (10, 5, 3, 0)*. D = 62, A = 60, B = 57, C = 55; the winner is D. This vector selects D, the Condorcet winner.

The results of five different vectors are collected in Table 7.4. All the computations are done with the same method: each group's contribution to each candidate is "number of voters × the points of that position."

Rule	s	Scores	Winner
Plurality	(1, 0, 0, 0)	A 6, C 4, D 3, B 0	A
2-approval (top two)	(1, 1, 0, 0)	B 9, D 7, A 6, C 4	B
Borda	(3, 2, 1, 0)	B 22, D 21, A 18, C 17	B
Veto (0 for last)	(1, 1, 1, 0)	B 13, D 11, C 9, A 6	B
Another vector	(10, 5, 3, 0)	D 62, A 60, B 57, C 55	D

Table 7.4: Five score vectors on the same profile.

Example 7.6

Veto, *s = (1, 1, 1, 0)*: a candidate's score is the number of voters who did not put them last. Those put last are in the last row of the table: G1 puts C last, G2 and G3 put A last, and G4 puts D last.

- *4 + 3 = 7* voters put A last; A = *13 − 7 = 6*.
- Nobody puts B last; B = 13.
- 4 voters put C last; C = 9.
- 2 voters put D last; D = 11.

The vector **(10, 5, 3, 0)**:

- A: 1st in G1 (*4 × 10 = 40*), 4th in G2 and G3 (0), 1st in G4 (*2 × 10 = 20*); A = 60.
- B: 3rd in G1 (*4 × 3 = 12*), 2nd in G2 (*4 × 5 = 20*), 2nd in G3 (*3 × 5 = 15*), 2nd in G4 (*2 × 5 = 10*); B = 57.
- C: 4th in G1 (0), 1st in G2 (40), 3rd in G3 (*3 × 3 = 9*), 3rd in G4 (*2 × 3 = 6*); C = 55.
- D: 2nd in G1 (*4 × 5 = 20*), 3rd in G2 (*4 × 3 = 12*), 1st in G3 (30), 4th in G4 (0); D = 62.

This table gives one of the chapter's main messages in numbers: with the same ballots, changing only the score vector changes the winner among A, B, and D. There is no mathematical answer as to which score vector is "right." Vectors that give a lot of weight to first place resemble plurality and promote polarizing candidates (A); vectors that leave equal gaps between positions promote compromise candidates (B).

Scoring rules share one good property: they are *monotone*. If a voter moves the winning candidate up one position on their ballot, that candidate receives either the same number of points or more, because in a score vector a higher position gives at least as many points as a lower one. The other candidate, with whom the moved candidate swapped places, receives at most the same number of points. The winner's margin over the other candidates therefore does not shrink, and the winner continues to win. This observation matters because instant-runoff voting does not satisfy this property (Section 7.3.7).

Common mistake

Mistaking instant-runoff voting for a scoring rule. IRV also uses a full ranking, but the vote counted for a candidate in a round depends on which candidates were eliminated earlier. The same ballot may count for A in one round and for C in the next. IRV therefore cannot be written with a single fixed score vector.

7.2.6 Instant-runoff voting

One problem with the two-round system is that it takes voters to the polls twice. Another is that only two candidates remain for the second round, so the chances of a candidate who came third in the first round end at once. **Instant-runoff voting** (IRV; also known as ranked-choice voting) looks for a solution to both problems at once: a full ranking is collected from each voter once, and the rounds are then held "instantly," by computer or by hand, on these rankings.

Definition 7.8

Each voter reports a full ranking. At the start all candidates are in the race. In each round:

1. Each ballot is counted for the highest-ranked candidate on that ballot who is *still in the race*.
2. If a candidate's vote count is more than half of the total votes, that candidate wins.
3. Otherwise the candidate with the fewest votes is eliminated; their ballots pass in the next round to the next candidate still in the race in the ranking.

Under this rule no ballot is wasted: if the voter's favorite is eliminated, the vote is transferred to the next candidate in the ranking. The difference from the two-round system is that candidates are eliminated one at a time and the votes are redistributed after each elimination.

Example 7.7

1. **Round 1:** candidates in the race are A, B, C, D. First choices are A = 6, B = 0, C = 4, D = 3. A majority needs 7; nobody reaches it.
2. **Elimination:** the candidate with the fewest votes is B (0 votes). B is eliminated. Since nobody put B first, there are no votes to transfer either.
3. **Round 2:** A, C, D are in the race. Each ballot is counted for the highest-ranked candidate in the race; B's elimination changes the candidate counted on no ballot. A = 6, C = 4, D = 3. Again there is no majority.
4. **Elimination:** the candidate with the fewest votes is D (3). D is eliminated. G3's ballot is D $\succ$ B $\succ$ C $\succ$ A; since D and B have been eliminated, the next candidate in the race is C. 3 votes pass from D to C.
5. **Round 3:** A and C are in the race. A = 6, C = $4 + 3 = 7$. $7 \geq 7$: C wins with a majority.

In this profile the result of IRV is the same as that of the two-round system: C. The reason is that after two candidates are eliminated, the remaining pair is the same as the pair that reaches the second round of the two-round system. In general, however, the two rules can give different results, because in IRV the elimination order can change as votes are transferred. Another notable point: IRV eliminates D, the Condorcet winner, in the second round. Because D's first-choice count is small, D drops out at an early stage of the race even though D could beat all rivals one on one.

Common mistake

Transferring an eliminated candidate's votes to the candidate ranked second on the ballot. The vote passes to the next candidate in the ranking who is *still in the race*. On G3's ballot B comes after D, but since B was eliminated earlier, the vote goes not to B but to C. If the candidate in second place has been eliminated, look at the third; if that one has also been eliminated, the fourth.

7.2.7 Condorcet: pairwise comparisons

The rules so far evaluate the candidates all together. Condorcet's proposal is different: every pair of candidates is compared separately, as if only those two were running. The contest between two candidates is decided by the number of voters who place one of them higher than the other.

Definition 7.9

Let $N(x, y)$ be the number of voters who prefer x to y. If $N(x, y) > N(y, x)$, then x **beats** y **by pairwise majority**.

- A candidate who beats all rivals is the **Condorcet winner**.
- A candidate who loses to all rivals is the **Condorcet loser**.

The idea of a Condorcet winner is very attractive. If the majority of voters prefer a candidate to each rival, then whoever is elected instead, a majority that wants this candidate in their place will be dissatisfied. But a Condorcet winner may not exist in every profile; this is the subject of Section 7.3.

For four candidates there are $4 \cdot 3 / 2 = 6$ pairs. In general, for m candidates the number of pairs is $m(m-1)/2$. The results are written in a table: cell (x, y) is $N(x, y)$, the number of voters who prefer x to y.

Example 7.8

For each pair the groups are examined one by one. The numbers are numbers of voters; in each row the two numbers must add up to 13.

1. **A – B:** those who say A ≻ B are G1 (4) and G4 (2); those who say B ≻ A are G2 (4) and G3 (3). 6 – 7: B wins.
2. **A – C:** A ≻ C: G1 (4), G4 (2); C ≻ A: G2 (4), G3 (3). 6 – 7: C wins.
3. **A – D:** A ≻ D: G1 (4), G4 (2); D ≻ A: G2 (4), G3 (3). 6 – 7: D wins.
4. **B – C:** B ≻ C: G1 (4), G3 (3), G4 (2); C ≻ B: G2 (4). 9 – 4: B wins.
5. **B – D:** B ≻ D: G2 (4), G4 (2); D ≻ B: G1 (4), G3 (3). 6 – 7: D wins.
6. **C – D:** C ≻ D: G2 (4), G4 (2); D ≻ C: G1 (4), G3 (3). 6 – 7: D wins.

x ╲ y	A	B	C	D	wins
A	—	6	6	6	0
B	7	—	9	6	2
C	7	4	—	6	1
D	**7**	**7**	**7**	—	**3**

Table 7.5: Pairwise comparison table. Cell (x, y) = number of voters who prefer x to y; wins = the number of entries in the row greater than 6.5.

Once Table 7.5 is complete, the result is clear: D beats all three rivals 7–6 and is the Condorcet winner. A loses to everyone 6–7 and is the Condorcet loser. What is surprising is that plurality picked exactly this Condorcet loser, A. A has the highest first-choice count, but all seven voters who do not put A first put A last; these seven voters support every candidate running against A.

The row sums of the table also say something: row A is $6 + 6 + 6 = 18$, row B is $7 + 9 + 6 = 22$, row C is $7 + 4 + 6 = 17$, and row D is $7 + 7 + 7 = 21$. These are exactly the Borda scores. This is a consequence of the interpretation in Section 7.2.3: the Borda score is the sum of the votes a candidate collects in all pairwise comparisons. The difference between Borda and Condorcet is also seen here. Condorcet looks only at who wins (is 7 greater than 6?), while Borda also counts the size of the margin. B's large 9–4 margin over C compensates in the Borda total for its small 6–7 loss to D.

7.2.8 The pairwise majority graph and Copeland

The results of pairwise comparisons can also be shown with a graph. In this graph each candidate is a vertex. If x beats y by pairwise majority, an arrow is drawn from x to y. This graph is called the **pairwise majority graph**.

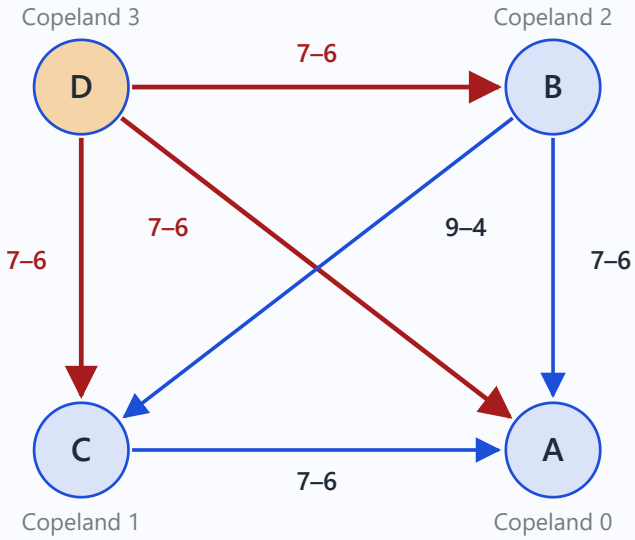


Figure 7.5: The pairwise majority graph of the example profile. An arrow x → y shows that a majority prefers x to y; the labels are vote counts. The three arrows leaving D are highlighted. Copeland scores are written next to the vertices.

When the number of voters is odd, there can be no tie between two candidates; so there is exactly one arrow between every pair. Such a graph is called a **tournament**; the name comes from league matches in which every team plays every other team once. In Figure 7.5 three arrows leave D and none enter D. The general rule is:

- **Condorcet winner** = the vertex that receives no arrows (D).
- **Condorcet loser** = the vertex that receives all arrows and gives none (A).

If every candidate receives at least one arrow, that is, if there is no vertex that receives none, there is no Condorcet winner. In this case the graph contains a cycle: starting from a candidate and following the arrows, one can return to the same candidate. A cycle that covers all candidates causes the absence of a Condorcet winner.

Definition 7.10

In the **Copeland rule** each candidate's score is the number of wins in pairwise comparisons, that is, the number of arrows leaving that candidate in the graph. The candidate with the highest score wins. (Where ties are possible, ties earn half a point.)

In the example profile the Copeland scores are D 3, B 2, C 1, A 0; these scores are the same as the last column of Table 7.5. Copeland gives the ranking D $\succ$ B $\succ$ C $\succ$ A, and the winner is D. A good property of the Copeland rule is that it always selects the Condorcet winner if there is one: the Condorcet winner receives the highest possible score with $m - 1$ wins, and no other candidate can reach this score, because every other candidate loses at least once (to the Condorcet winner). If there is no Condorcet winner, Copeland still gives a ranking, but ties are frequent in this ranking.

7.2.9 The same profile, four different winners

So far the votes of the same 13 voters have been counted in seven different ways. The results are put side by side in Table 7.6. The Kemeny row of the table is computed in Section 7.3.4.

Rule	Count	Winner
Majority	A 6 < 7	none
Plurality	A 6, C 4, D 3, B 0	A
Two-round system	C 7 – A 6	C
Instant-runoff	C 7 – A 6 (round 3)	C
Borda	B 22, D 21, A 18, C 17	B
2-approval (top two)	B 9, D 7, A 6, C 4	B
Condorcet	D 3 wins	D
Kemeny	D $\succ$ B $\succ$ C $\succ$ A (44 agreements)	D

Table 7.6: The same profile, different rules.

Two lessons can be drawn from the table. First: "the will of the people" depends on the rule. Although the voters' ballots do not change, the winner can be A, B, C, or D when the rule changes; that is, the outcome is determined by the rule as much as by the voters. Which rule will be used must therefore be decided explicitly before the vote. Second: some rules can give counterintuitive results. Plurality can select A, the Condorcet loser that loses to everyone; IRV can eliminate D, the Condorcet winner that beats everyone, early.

Why each rule picks a different candidate is explained by the features of the profile noted in Section 7.1.3. Because plurality looks only at first place, it picks A, who has the most “loyal” voters. The two-round system and IRV hold a final contest among the candidates with strong first choices and pick C. Borda and 2-approval, because they also count second places, pick the compromise candidate B. Condorcet and Kemeny, because they look at pairwise contests, pick D, who wins every pairwise contest.

7.2.10 Code trace: borda()

Counting under voting rules can be done with programs of a few lines. The short Python function below does the Borda count. The function’s input is a list called `ballots`. Each element of the list is a pair of the form `(count, order)`: `count` is the number of voters who gave that ranking, and `order` is a string that lists the candidates from most preferred. The example profile is written as follows:

```
ballots = [(4, "ADBC"), (4, "CBDA"), (3, "DBCA"), (2, "ABCD")]
```

```
def borda(ballots):
  score = {}
  for count, order in ballots:
    m = len(order)
    for k, cand in enumerate(order):
      p = count * (m - 1 - k)
      score[cand] = score.get(cand, 0) + p
  winner = max(score, key=score.get)
  return winner, score
```

The code works as follows:

- `score` is an empty dictionary; each candidate’s accumulated score is kept here.
- The outer loop visits each voter group once: `count` is the size of the group and `order` is the group’s ranking.
- `m = len(order)` is the number of candidates; here 4.
- The inner loop visits the candidates in the ranking together with their positions. `enumerate` gives the position to `k`, starting from 0: first place is `k = 0` and last place is `k = m − 1`.
- `p = count * (m - 1 - k)` is the group’s contribution to this candidate. For `k = 0` the score is *m − 1 = 3*, and for `k = 3` it is 0. This is the point scale of the definition written with an index that starts from 0.
- `score.get(cand, 0)` returns 0 if the candidate is not yet in the dictionary; a candidate seen for the first time thus starts from 0.
- `max(score, key=score.get)` returns the key (candidate) with the largest score.

Table 7.7 traces the inner loop of the first group step by step and the other groups in bulk.

group	k	candidate	p	score (A, B, C, D)
4 × ADBC	0	A	4 × (3 − 0) = 12	12, –, –, –
4 × ADBC	1	D	4 × (3 − 1) = 8	12, –, –, 8
4 × ADBC	2	B	4 × (3 − 2) = 4	12, 4, –, 8
4 × ADBC	3	C	4 × (3 − 3) = 0	12, 4, 0, 8
4 × CBDA	0–3	C, B, D, A	12, 8, 4, 0	12, 12, 12, 12
3 × DBCA	0–3	D, B, C, A	9, 6, 3, 0	12, 18, 15, 21
2 × ABCD	0–3	A, B, C, D	6, 4, 2, 0	18, 22, 17, 21

Table 7.7: Trace of borda(). The last column is the score dictionary after the step.



ballots: row = voter group; 1st place 3 points … 4th place 0

Figure 7.6: The final state of the borda() trace. On the left the four input groups, on the right the accumulated scores; the longest bar is B's (22).

The function returns `("B", {"A": 18, "D": 21, "B": 22, "C": 17})`. Figure 7.6 shows this final state. The candidate with the highest score is B (22), and no voter put B first.

One subtlety of the code is the tie case. If two candidates receive the same highest score, `max` returns the one it saw first in the dictionary. For a voting rule this is an arbitrary choice: the outcome is determined by the order in which candidates entered the dictionary, that is, by the order of the input list. Real rules need instead a **tie-breaking rule** announced in advance: drawing lots, a fixed order of candidates, or a second criterion (for example the first-choice count).

In terms of complexity, the function processes every candidate once for each group. With n voter groups and m candidates, the inner loop runs $n \cdot m$ times in total; the running time is $O(nm)$. Counting is easy; what is hard is understanding which properties the rule satisfies.

7.2.11 Code trace: irv()

The code for instant-runoff voting is a little longer than Borda's, because the count is repeated in rounds. The function takes the `ballots` list and the list of candidates.

```
def irv(ballots, cands):
  remaining = list(cands)
  while True:
    tally = {a: 0 for a in remaining}
    for count, order in ballots:
      top = next(a for a in order if a in remaining)
      tally[top] += count
    leader = max(tally, key=tally.get)
    if 2 * tally[leader] > sum(tally.values()):
      return leader
    loser = min(tally, key=tally.get)
    remaining.remove(loser)
```

The code works as follows:

- `remaining` is the list of candidates still in the race; in a tie, the order in the candidate list decides (if a set were used, the order could change from run to run); at the start all candidates are in the race.
- Each pass of the `while True` loop is one round. The loop is exited only through `return`, that is, when a winner is found.
- `tally` is a counter starting from 0 for each candidate in the race. The counters are reset in every round; the votes are counted from scratch each round.
- `next(a for a in order if a in remaining)` scans the ranking from the beginning and returns the first candidate *still in the race*. The transfer of votes happens by itself in this line: eliminated candidates are skipped because they are not in `remaining`.
- `leader` is the candidate with the most votes. The condition `2 * tally[leader] > sum(tally.values())` tests whether the leader's votes exceed half of the total. Multiplying both sides by 2 avoids dealing with fractional thresholds such as 6.5.
- If there is no majority, `min(tally, key=tally.get)` finds the candidate with the fewest votes and `remaining.remove` takes them out of the race.

round	remaining	tally	leader	condition	result
1	A, B, C, D	A 6, B 0, C 4, D 3	A	12 ≤ 13	B eliminated
2	A, C, D	A 6, C 4, D 3	A	12 ≤ 13	D eliminated
3	A, C	A 6, C 7	C	14 > 13	C returned

Table 7.8: Trace of irv() (13 votes in total; majority condition 2 · tally > 13).

In the first round the four ballot groups are processed in turn: for the group ADBC the first candidate in the race is A and `tally[A] += 4`; for CBDA, `tally[C] += 4`; for DBCA, `tally[D] += 3`; for ABCD, `tally[A] += 2`. The result is A 6, B 0, C 4, D 3. The leader is A, but since $2 \times 6 = 12 \le 13$, there is no majority. B (0), with the fewest votes, is eliminated. In the second round the counters are reset and

the count is repeated; since nobody put B first, the numbers do not change. D (3) is eliminated. In the third round the scan of the DBCA group skips D and B and stops at C: `tally[C] += 3`. This is the transfer of D's votes to C. Since *2 × 7 = 14 > 13*, C is returned.

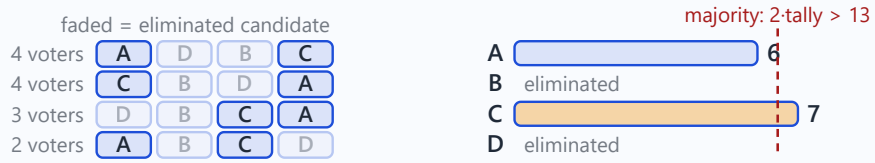


Figure 7.7: The last round of the irv() trace. Faded cells are eliminated candidates; each group's vote is counted for the first candidate in the race. A has 6, C has 7; C satisfies the majority condition.

Common mistake

- **Using "≥ half" as the threshold:** with an even number of voters, a candidate with exactly half of the votes does not have a majority. The condition is strict inequality: `2 * tally > total`.
- **Not thinking about ties:** if the two candidates with the fewest votes are tied, `min` picks one of them. The code eliminates the one that comes first in the `remaining` list; this result is reproducible but arbitrary, because it depends on the order of the candidate list. In a real count a tie must be broken by a rule announced in advance.

In each round, for n groups the ranking is scanned in at most m steps, so one round takes $O(nm)$. Since there are at most $m - 1$ rounds, the total time is $O(nm^2)$. This is also a fast computation; the difficulties of IRV lie not in the counting but in its properties.

7.3 The Condorcet paradox

In the example profile there is a Condorcet winner: D. But not every profile has such a candidate. If there is no such candidate, the order of the votes can determine the outcome. The Kemeny rule produces a social ranking in this case too.

7.3.1 The Condorcet cycle

Consider the following profile with nine voters and three candidates: 4 voters have A ≻ B ≻ C, 3 voters have B ≻ C ≻ A, and 2 voters have C ≻ A ≻ B. Each voter's ranking is consistent by itself, that is, transitive.

rank	4 voters	3 voters	2 voters
1.	A	B	C
2.	B	C	A

rank	4 voters	3 voters	2 voters
3.	C	A	B

Table 7.9: The cycle profile: 9 voters, 3 candidates.

Example 7.9

1. **A – B:** those who prefer A to B are column 1 (4) and column 3 (2): *4 + 2 = 6*. Those who prefer B are column 2: 3. The majority is A ≻ B (6 – 3).
2. **B – C:** those who prefer B are *4 + 3 = 7*, and those who prefer C are 2. The majority is B ≻ C (7 – 2). Expectation of transitivity: since A ≻ B and B ≻ C, we should have A ≻ C.
3. **C – A:** those who prefer C to A are column 2 (3) and column 3 (2): *3 + 2 = 5*. Those who prefer A are 4. The majority is C ≻ A (5 – 4)!

Result: A ≻ B ≻ C ≻ A. The social preference is cyclic. For every candidate there is a majority that beats them; there is no Condorcet winner.

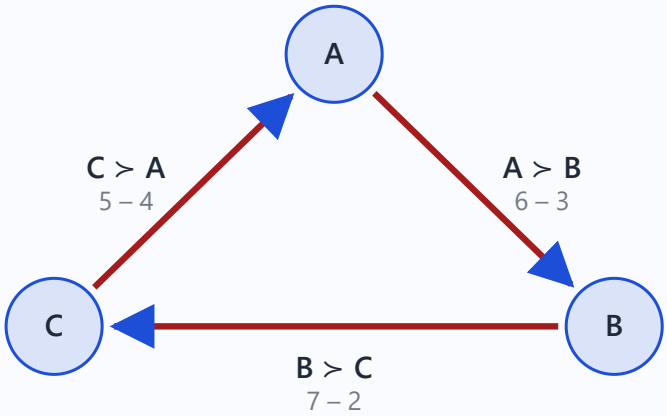


Figure 7.8: The Condorcet cycle. A beats B (6 – 3), B beats C (7 – 2), and C beats A (5 – 4). The arrows form a circle; no vertex is left without an incoming arrow.

Figure 7.8 shows the situation with the pairwise majority graph. The three arrows form a circle: an arrow enters every vertex, so there is no Condorcet winner. This phenomenon is called the **Condorcet paradox** or **Condorcet cycle**.

Definition 7.11

If the pairwise majority relation contains a sequence of candidates in which x_1 beats x_2, x_2 beats x_3, ..., and x_k beats x_1, the profile has a **Condorcet cycle**. In this case the pairwise majority relation is not transitive.

The importance of the paradox is this: individuals being consistent does not guarantee that the community is consistent. The "majority" in the sentences "the majority prefers A to B" and "the majority prefers B to C" is not the same set of people. The first majority is made up of columns 1 and 3, the second of columns 1 and 2. Different majorities make different decisions, and when these decisions are put together, a cycle can arise.

Common mistake

Thinking that the cycle comes from one of the voters being inconsistent. All three rankings in the profile are transitive. The cycle arises from *aggregating* consistent rankings by majority rule. Another mistake is to count the candidate with "the most pairwise wins" as the Condorcet winner when there is a cycle: in a cycle every candidate has one win, and by definition there is no Condorcet winner.

7.3.2 The agenda determines the winner

The cycle has a practical consequence. Many committees and parliaments make decisions by **sequential pairwise voting**: first two options are voted on, the winner meets the next option, and this continues until the options run out. The order in which the options are voted on is called the **agenda**. In the cycle profile three different agendas are possible (one chooses which pair is voted on first; the remaining candidate is left for last).

Example 7.10

1. **Agenda 1: first A – B, then C.** A – B: 6 – 3, A wins. A – C: 4 – 5, C wins. Winner: C.
2. **Agenda 2: first B – C, then A.** B – C: 7 – 2, B wins. B – A: 3 – 6, A wins. Winner: A.
3. **Agenda 3: first A – C, then B.** A – C: 4 – 5, C wins. C – B: 2 – 7, B wins. Winner: B.

All three agendas produce different winners.

Agenda	First vote	Winner
(A – B), then C	A	C
(B – C), then A	B	A
(A – C), then B	C	B

Table 7.10: Three agendas in the cycle profile.

The pattern in the example is a general rule: in a three-candidate cycle, *the candidate left for last wins*. The reason is this: in the first vote two candidates meet, and the winner meets the candidate left for last. In a three-candidate cycle each candidate beats exactly one candidate and loses to exactly one.

The winner of the first vote has beaten the only candidate it can beat in the first vote; so the only candidate it loses to is the one left for last, and it loses the final vote. For example, in agenda 1, A beats B; the candidate in the cycle who beats A is C, and C is at the end.

This gives great power to the person who sets the agenda (for example a committee chair). The chair can choose the outcome by putting the candidate they want to win last; and each vote is itself an honest majority vote. This situation is called **agenda manipulation**. In a profile with a Condorcet winner, however, this power does not exist: whichever order it is voted in, the Condorcet winner beats every candidate it meets and wins in the end.

7.3.3 Back to the game: how common are cycles?

The two questions of the game at the start of the chapter can now be answered. The profile at the game's first level (A ≻ B ≻ C, B ≻ C ≻ A, C ≻ A ≻ B) is a cycle: A – B 2 – 1, B – C 2 – 1, C – A 2 – 1. The social preference is cyclic; there is no Condorcet winner. The answer to the first question is "no": even if individuals are consistent, the community does not have to be. The answer to the second question is "yes": the person who sets the agenda chooses the outcome by leaving the candidate they want to win for last. The results of the agendas follow the same pattern as in Table 7.10: C wins under the agenda (A – B) then C, A wins under (B – C) then A, and B wins under (A – C) then B.

A natural question is how often cycles occur. For this a probability model is needed. The simplest model is that each voter chooses one of the six possible rankings with equal probability, independently of the others. For three voters and three candidates the number of possible profiles is $6 \times 6 \times 6 = 6^3 = 216$. The number of these profiles containing a cycle can be counted: a cycle arises only when the voters' rankings are an arrangement of the triple A ≻ B ≻ C, B ≻ C ≻ A, C ≻ A ≻ B or of its reverse, the triple A ≻ C ≻ B, C ≻ B ≻ A, B ≻ A ≻ C. Each triple can be distributed among three voters in $3! = 6$ different ways; there are $2 \times 6 = 12$ profiles with cycles in total. The probability is $12/216 \approx 0.056$, that is, about 5.6%.

As the number of voters grows very large, in the same random model the probability of a cycle with three candidates rises to about 8.8% (Guilbaud, 1952). As the number of candidates grows, this rate rises quickly too. These numbers may look alarming, but in real election and survey data cycles are seen less often than this "random preference" calculation predicts (Regenwetter et al., 2006). The reason is that real preferences are not random: people's preferences resemble one another and are often organized along a common axis (for example left–right). Why cycles do not arise with such orderly preferences is explained in Section 7.5.6.

Common mistake

Interpreting rates such as 5.6% or 8.8% as the frequency of cycles in real elections. These numbers are a consequence of the assumption that "every ranking is equally likely." When the assumption changes, the number changes too; real data show lower rates.

7.3.4 The Kemeny rule

When there is a cycle, the Condorcet idea cannot give a winner. A decision still has to be made. John Kemeny (1959) proposed to solve this problem as follows: choose the social ranking that *agrees most* with the voters' pairwise preferences.

Definition 7.12

The **agreement score** of a ranking σ of the candidates is the sum, over every pair (x, y) in which x comes before y in σ, of the number of voters who prefer x to y:

$$agree(\sigma) = \Sigma_{x \text{ before } y \text{ (in } \sigma)} N(x, y).$$

The **Kemeny rule** chooses the ranking with the largest agreement score.

The agreement score has another interpretation. Between a voter's ranking and σ there is agreement on some pairs and disagreement on others. The agreement score is the sum, over all voters, of the number of pairs they agree on. Maximizing agreement is the same as minimizing the number of pairs they disagree on (the total "distance" between the rankings). In this sense the Kemeny ranking is the ranking "closest on average" to the voters' rankings.

Example 7.11

In the main profile, the agreement score of the ranking D $\succ$ B $\succ$ C $\succ$ A is read off Table 7.5. The six pairs in this ranking and the relevant numbers:

- D before B: $N(D, B) = 7$; D before C: 7; D before A: 7.
- B before C: $N(B, C) = 9$; B before A: 7.
- C before A: $N(C, A) = 7$.

Agreement: $7 + 7 + 7 + 9 + 7 + 7 = 44$. In this ranking every pair points in the same direction as that pair's pairwise majority; each pair's contribution is the largest possible value. No other ranking can exceed 44; when all 24 rankings are tried, the closest competitors (D $\succ$ B $\succ$ A $\succ$ C and B $\succ$ D $\succ$ C $\succ$ A) stay at 43.

The example shows a general observation: if the pairwise majority relation is transitive (there is no cycle), the ranking that follows this relation receives the largest contribution on every pair and is the Kemeny ranking. In particular, if there is a Condorcet winner, it is placed first in the Kemeny ranking. The reason is this: if the Condorcet winner is not first in a ranking, swapping it with the candidate immediately above raises that pair's contribution from the smaller number to the larger and changes no other pair; this raises the agreement.

In a cycle too, Kemeny makes a decision. In a cycle profile the three pairwise majorities cannot all be satisfied together; at least one must be violated. Kemeny violates the majority whose violation loses the least agreement, that is, the *weakest* majority.

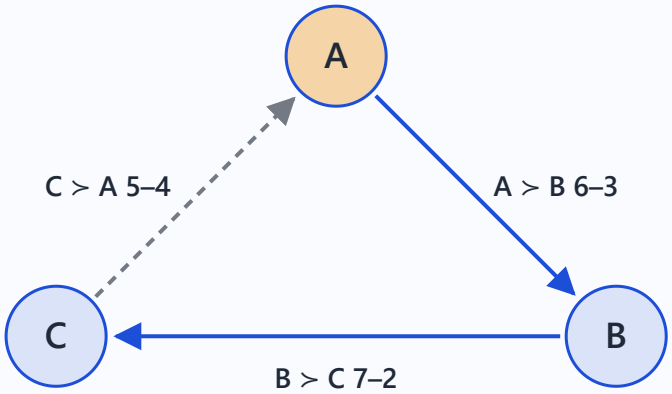


Figure 7.9: The Kemeny ranking in the cycle profile. The dashed arrow is the weakest majority, which Kemeny overrides ($C \succ A$, 5 – 4). The remaining two majorities give the ranking $A \succ B \succ C$.

Figure 7.9 shows the three edges of the cycle and the vote counts: $A \succ B$ 6 – 3, $B \succ C$ 7 – 2, $C \succ A$ 5 – 4. The edge with the narrowest margin is $C \succ A$. When this edge is overridden, the remaining two edges give the ranking $A \succ B \succ C$. The agreement of this ranking is $N(A, B) + N(A, C) + N(B, C) = 6 + 4 + 7 = 17$. A full comparison of the six rankings is in the code trace in Section 7.3.5.

The Kemeny rule has strong properties, but it also has a price: finding the best ranking is NP-hard (Bartholdi, Tovey, and Trick, 1989). For an NP-hard problem, all known algorithms take time that grows exponentially with the size of the input in the worst case. With few candidates all rankings can be tried, but with 15–20 candidates this is no longer practical. An important application of the Kemeny rule is **rank aggregation** (Dwork et al., 2001): merging the result lists of several search engines into one list, or combining the rankings of contest juries and committees.

7.3.5 Code trace: kemeny()

For a small number of candidates, the Kemeny ranking can be found by trying all rankings. The function below does exactly this. The function uses the `permutations` function from Python's standard library; the line `from itertools import permutations` must therefore appear at the start of the program. `permutations(cands)` generates all orderings of the candidates one by one.

```
def kemeny(ballots, cands):
  best, best_score = None, -1
  for order in permutations(cands):
    score = 0
    for i, x in enumerate(order):
      for y in order[i + 1:]:
        score += sum(n for n, b in ballots
                     if b.index(x) < b.index(y))
    if score > best_score:
      best, best_score = order, score
  return best, best_score
```

The code works as follows:

- `best` holds the best ranking found so far and `best_score` its agreement score. The initial value is *−1*; the first ranking tried (with a score of at least 0) is thus always accepted as the best.
- The outer loop tries every ranking `order`.
- The two inner loops run over all pairs *(x, y)* in the ranking; `order[i + 1:]` gives only the candidates that come after `x`. Each pair is thus handled once, in the direction "`x` first in the ranking."
- `b.index(x) < b.index(y)` tests whether that voter group put `x` before `y`. The sizes `n` of the groups satisfying the condition are summed; this is *N(x, y)*.
- If a ranking's score is strictly greater than the previous best, the best is updated. Among rankings with equal scores, the first one found is kept.

The trace for the cycle profile `ballots = [(4, "ABC"), (3, "BCA"), (2, "CAB")]` is given in Table 7.11. For three candidates there are *3! = 6* rankings, and each ranking contains *3 · 2 / 2 = 3* pairs.

order	pair 1	pair 2	pair 3	score	best
A ≻ B ≻ C	A–B: 6	A–C: 4	B–C: 7	**17**	A ≻ B ≻ C (17 > −1)
A ≻ C ≻ B	A–C: 4	A–B: 6	C–B: 2	12	unchanged
B ≻ A ≻ C	B–A: 3	B–C: 7	A–C: 4	14	unchanged
B ≻ C ≻ A	B–C: 7	B–A: 3	C–A: 5	15	unchanged
C ≻ A ≻ B	C–A: 5	C–B: 2	A–B: 6	13	unchanged
C ≻ B ≻ A	C–B: 2	C–A: 5	B–A: 3	10	unchanged

Table 7.11: Trace of kemeny() (cycle profile). In the pair columns, N(x, y) in the order of the ranking.

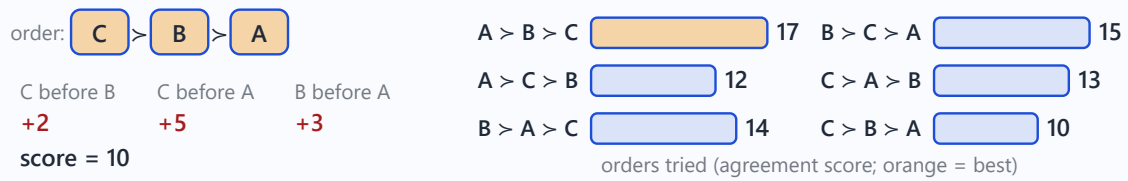


Figure 7.10: The last step of the kemeny() trace. At the top, the last ranking tried, C ≻ B ≻ A, and the contributions of its pairs (2 + 5 + 3 = 10); at the bottom, the agreement scores of the six rankings, with the best, A ≻ B ≻ C (17), in orange.

The function returns `(("A", "B", "C"), 17)`. The Kemeny ranking is A ≻ B ≻ C, and only the weakest majority, C ≻ A (5 – 4), has been overridden. The second-best ranking is B ≻ C ≻ A (15); this ranking overrides the majority A ≻ B (6 – 3) and loses 2 more points of agreement. A symmetry of the table is also notable: adding the score of a ranking and the score of its reverse always gives $3 \times 9 = 27$ (17 + 10, 12 + 15, 14 + 13). The reason is that on every pair, a voter agrees with exactly one of the two rankings.

The cost of this brute-force method grows quickly. For m candidates, $m!$ rankings are tried; each ranking has $m(m-1)/2$ pairs, and for each pair n groups are scanned. The total work is of the order $m! \cdot m(m-1)/2 \cdot n$. For 4 candidates there are 24 rankings, for 10 candidates 3,628,800, and for 20 candidates about 2.4×10^{18}. That Kemeny is NP-hard says that there is no known general way to escape this exponential growth.

7.3.6 Which criteria do the rules satisfy?

Different rules choose different winners. Criteria are needed to argue about which is "better." The criteria used to compare rules in social choice theory are also called *axioms*. Four basic criteria are defined below, and whether the rules satisfy them is collected in Table 7.12.

Definition 7.13

- **Condorcet consistency:** if there is a Condorcet winner, the rule selects it.
- **Condorcet loser criterion:** the rule never selects the Condorcet loser.
- **Monotonicity**: if a winning candidate is moved up on some voters' ballots (without changing the order of the other candidates relative to each other), it still wins. In short: supporting the winner should not make it lose.
- **Strategy-proofness**: no voter can obtain a better outcome, according to their true preference, by reporting a ranking different from their true preference.

Rule	Selects the Condorcet winner	Never selects the Condorcet loser	Monotone	Strategy-proof
Plurality	no	no (A)	yes	no

Rule	Selects the Condorcet winner	Never selects the Condorcet loser	Monotone	Strategy-proof
Borda	no (B)	yes	yes	no
Instant-runoff	no (C)	yes	no	no
Approval	no	no	yes	no
Condorcet (Copeland)	yes	yes	yes	no
Kemeny	yes	yes	yes	no

Table 7.12: Rules and criteria. The candidate in parentheses shows the violation in the example profile.

The table can be read row by row. In the example profile, plurality selects not the Condorcet winner D but A; A is also the Condorcet loser, so plurality violates both Condorcet criteria. Borda selects B instead of the Condorcet winner D. But Borda never selects the Condorcet loser. The reason is the observation in Section 7.2.7: the Borda score is the sum of the votes received in pairwise comparisons. The Condorcet loser gets fewer than half of the votes in every comparison; its Borda score therefore stays below the average Borda score of the candidates. In the example profile the average is *78 / 4 = 19.5*, and A's score is 18. A candidate below the average cannot get the highest score.

IRV eliminates the Condorcet winner D in the second round and selects C. IRV does not select the Condorcet loser: in the last round two candidates face each other, and the Condorcet loser loses this last contest. But IRV is not monotone (Section 7.3.7). In approval voting, since the outcome depends on approval thresholds, no Condorcet criterion can be guaranteed. Copeland and Kemeny, being based on pairwise comparisons, satisfy both Condorcet criteria.

All the "no" entries in the last column have a common cause: with three or more candidates, no reasonable rule other than a dictatorship is strategy-proof. This general result is the Gibbard–Satterthwaite theorem and is treated in Section 7.5.4.

A Condorcet winner does not exist in every profile; the plain Condorcet rule cannot find a winner in a cycle. Copeland may also fail to decide in a cycle: the scores come out equal (for example in the cycle profile all three candidates have one win each) and a tie-breaking rule is needed. One of the rules that remedies this deficiency is maximin.

Definition 7.14

In the **maximin** (Simpson) rule, for each candidate the *lowest* vote count received in pairwise comparisons is found: $score(x) = \min_{y \neq x} N(x, y)$. The candidate with the highest score wins.

Example 7.12

In the main profile the smallest number in each row of Table 7.5 is taken:

- A: *min(6, 6, 6) = 6*.
- B: *min(7, 9, 6) = 6*.
- C: *min(7, 4, 6) = 4*.
- D: *min(7, 7, 7) = 7*.

The winner is D. In the cycle profile, A's worst comparison is 4 against C, B's is 3 against A, and C's is 2 against B; maximin selects A. This is the same as the candidate at the head of the Kemeny ranking.

Maximin selects the Condorcet winner if there is one: the Condorcet winner gets more than half of the votes in every comparison, while every other candidate gets fewer than half in at least one comparison (against the Condorcet winner). It decides in a cycle too: the winner is the candidate whose worst defeat is the mildest.

7.3.7 Monotonicity violation in instant-runoff voting

Monotonicity seems to be one of the most basic properties one would expect of a voting rule: giving more support to a candidate should not hurt them. Scoring rules always satisfy this property (Section 7.2.5). IRV does not. An example with 17 voters shows this.

rank	6 voters	6 voters	5 voters
1.	A	B	C
2.	B	C	A
3.	C	A	B

Table 7.13: A monotonicity example: 17 voters, 3 candidates (9 votes are needed for a majority).

Example 7.13

Before the campaign (Table 7.13):

1. Round 1: A = 6, B = 6, C = 5. There is no majority (9); C, with the fewest votes, is eliminated.
2. C's 5 votes pass to the next candidate, A. Round 2: A = *6 + 5 = 11*, B = 6. A wins.

After the campaign: A campaigns, and 2 of the 6 voters who said B $\succ$ C $\succ$ A put A first; they now say A $\succ$ B $\succ$ C. The profile: 8 voters A $\succ$ B $\succ$ C, 4 voters B $\succ$ C $\succ$ A, 5 voters C $\succ$ A $\succ$ B. A's support has only increased; nothing else has changed.

1. Round 1: A = 8, B = 4, C = 5. There is no majority; this time B, with the fewest votes, is eliminated (not C).
2. B's 4 votes pass to the next candidate, C. Round 2: C = *5 + 4 = 9*, A = 8. C wins.

A lost because it gained more support: IRV is not monotone.

The mechanism of the violation is this. In IRV, whether a candidate wins depends not only on its own votes but also on *whom* it will face in the last round. In the first profile A faces B in the last round, and since C's voters prefer A to B, A wins. The 2 voters who moved to A in the campaign were taken from B; B's first-choice count falls from 6 to 4, and B drops below C. The elimination order changes: this time B is eliminated and A faces C in the last round. Since B's voters prefer C to A, C wins. Moving A up changed the elimination order.

The pairwise comparisons in this profile are interesting too: in the first profile A beats B 11 – 6, B beats C 12 – 5, and C beats A 11 – 6. The profile is a Condorcet cycle. Monotonicity violations are mostly seen in profiles like this, where there is no Condorcet winner or the race is very close.

Common mistake

Confusing a monotonicity violation with strategic voting. The 2 voters in the example do not lie; they really change their mind and begin to support A more. The problem is that the rule reacts perversely to this stronger support. Strategic voting, by contrast, is the voter reporting a ballot different from their true preference.

7.4 Strategic voting

So far every voter has been assumed to write their true preference on the ballot. But as noted in Section 7.1.2, the ballot is a strategy. A voter who knows how the rule works and how the other voters will vote can write a ballot different from their true preference and obtain a better outcome according to their true preference. This is called **strategic voting** or **manipulation**. The example profile shows two kinds of strategic voting with numbers.

7.4.1 Tactical voting under plurality

In honest voting the plurality winner is A: A 6, C 4, D 3, B 0. This is the outcome that bothers Group 3 (D $\succ$ B $\succ$ C $\succ$ A) most, because A is their worst candidate.

Example 7.14

1. **Reading the situation:** Group 3's favorite D gets only 3 votes; D has no chance of winning. The only candidate who can stop A is C, who is second with 4 votes. Group 3 prefers C to A (in their ranking C is third and A fourth).
2. **The strategic ballot:** Group 3 votes for C instead of D; that is, writes C $\succ$ D $\succ$ B $\succ$ A on the ballot.

3. **The new count:** A 6, C *4 + 3 = 7*, B 0, D 0. C wins.
4. **The outcome for the group:** according to their true preference D ≻ B ≻ C ≻ A, C is better than A. Lying has improved this group's situation.

This behavior is called **tactical voting**, or the "do not waste your vote" behavior: the voter votes not for their favorite but for the candidate who has a chance of winning and whom they find less bad. The cause of tactical voting is **vote splitting**: because the votes of the seven voters who oppose A were split between C and D, A was able to win with the support of a minority of voters.

Tactical voting also has a long-term consequence. As voters think that voting for small candidates is wasted, votes concentrate on the two strongest candidates. In political science this observation is called **Duverger's law**: a single-round plurality system pushes toward a two-party political structure. Single-member districts in the United Kingdom and the United States are frequently cited examples of this law.

7.4.2 Burying under Borda

The Borda count uses more information and is less affected by vote splitting. But it too is open to strategic behavior, and in a different way. In the honest Borda count B = 22 and D = 21. Group 3's favorite D loses by only 1 point.

Example 7.15

1. **Manipulation:** the 3 voters of Group 3 push the main rival B from 2nd place to 3rd. While the true ranking is D ≻ B ≻ C ≻ A, the ballot becomes D ≻ C ≻ B ≻ A.
2. **The point difference:** for each manipulator, B gets 1 point instead of 2 and C gets 2 points instead of 1. For three voters, B *−3* and C *+3*. A's and D's scores do not change.
3. **New totals:** A 18, B *22 − 3 = 19*, C *17 + 3 = 20*, D 21.
4. **Result:** D wins. Group 3 got its favorite elected by lying.

candidate	honest	change	strategic
A	18	0	18
B	**22**	−3	19
C	17	+3	20
D	21	0	**21**

Table 7.14: Burying under Borda: honest and strategic counts.

This manipulation is called **burying**: the voter "buries" the favorite's main rival lower on their ballot than its true position. Under Borda, burying is very effective, because every difference in rank turns directly into a difference in points. Pushing a rival to the very bottom reduces the points given to it as

much as possible.

Burying has a risk: it can backfire. Pushing a rival down raises the score of another candidate who moves up in its place. If this other candidate is worse for the burying voters and rises enough, that candidate wins. For example, look at this profile with 9 voters: $4 \times A \succ B \succ C$, $3 \times C \succ B \succ A$, $2 \times B \succ C \succ A$. The honest Borda (2, 1, 0) scores are A 8, B 11, C 8, and B wins. If the A voters bury B and report $A \succ C \succ B$, C gains 4 points and B loses 4: A 8, B 7, C 12, and C wins. C is the worst candidate for the A voters; burying has made their situation worse. A successful burying requires accurate information about the other voters' ballots.

Common mistake

Measuring the success of a manipulation by whether the manipulator's favorite wins. The criterion is whether the outcome improves according to the manipulator's *true* preference. In the plurality example Group 3's favorite D does not win, but since the outcome moves from A to C, the group is better off; this too is a successful manipulation.

7.4.3 How much does strategic voting disrupt the system?

The effect of strategic behavior differs by rule:

- **Plurality:** vote splitting is the basic problem. Two similar candidates share the votes and a third wins. Tactical voting is common and simple; the voter only needs to know who is ahead.
- **Borda:** burying the rival at the bottom is very effective. If everyone buries, the reported rankings drift away from the voters' true preferences, and the outcome can be a candidate nobody really wants.
- **Instant-runoff:** computing a manipulation is harder but possible; moreover the rule is not monotone. "IRV is immune to strategic voting" is a widespread but wrong belief.
- **The computational barrier:** for some rules (for example IRV and a variant of Copeland), finding a profitable manipulation is NP-hard (Bartholdi et al., 1989–91). This difficulty makes manipulation not impossible, but computationally expensive in the worst case.

The effect of strategic voting depends on the distribution of preferences. In close races the strategic behavior of a few voters can change the winner; in races with large margins, the manipulation of small groups has no effect on the outcome. Transparent rules and clear information reduce some manipulations, but no general rule can eliminate strategic behavior in every case. This last sentence is not an observation but a theorem (Section 7.5.4).

7.5 Impossibility theorems

In the previous sections a flaw was found in every rule: plurality can select the Condorcet loser, Borda is open to manipulation, IRV is not monotone, a Condorcet winner does not exist in every profile, and in a cycle a rule for a Condorcet extension (maximin, Kemeny) is needed. The natural question is: is there a more cleverly designed rule free of all these flaws? The answer of the theorems in this section, when there are three or more candidates, is "no." These results teach not to give up the search for rules but to choose consciously which property to give up.

7.5.1 Arrow's conditions

Kenneth Arrow (1951) asked the question in a very general form. Consider a rule that produces a social ranking, that is, a social welfare function (SWF). There are four conditions one might expect of such a rule that look very reasonable at first sight.

Definition 7.15

For an SWF F:

- **Unrestricted domain**: every preference profile is admitted and the outcome is always a complete and transitive ranking.
- **Pareto (unanimity)**: if everyone prefers x to y, society prefers x to y too.
- **Independence of irrelevant alternatives (IIA)**: the social order of x and y depends only on the voters' x–y order. If in two profiles every voter orders x and y the same way, the social ranking also orders x and y the same way in the two profiles.
- **Non-dictatorship**: there is no single voter whose preference is society's preference in every profile.

Why each condition is reasonable can be explained briefly as follows. Unrestricted domain asks that the rule not restrict in advance what voters may think and that it give a consistent outcome in every case; the Condorcet cycle breaks the "transitive outcome" part of this condition for the majority ranking. Pareto is respect for unanimity: if all voters want the same thing, there is no reason for society to decide otherwise. IIA asks that the decision between two candidates not be affected by the presence or position of a third; for example, it would be odd for someone who says "between fish and chicken I would pick fish" to switch to chicken when pasta is added to the menu. The non-dictatorship condition is the most basic expectation of fairness.

Theorem 7.1 (Arrow, 1951)

If there are at least 3 candidates, there is no social welfare function that satisfies unrestricted domain, Pareto, IIA, and non-dictatorship together. In other words, every SWF that satisfies unrestricted domain, Pareto, and IIA is a dictatorship.

The proof of the theorem is outside the scope of these notes; readable proofs can be found in the sources in Further reading. The theorem does not hold for two candidates: with two candidates, simple majority satisfies all four conditions. This case is the subject of May's theorem in Section 7.5.5.

Common mistake

Understanding IIA as "if a non-winning candidate withdraws, the winner should not change." This is a common but incomplete interpretation. Arrow's IIA can be violated with the candidate set fixed: a change in the social order of x and y when voters change only the positions of *other* candidates is also an IIA violation. The example in Section 7.5.2 shows both kinds.

7.5.2 Borda and irrelevant alternatives

The Borda count satisfies unrestricted domain (in every profile the scores are added and a ranking comes out), satisfies Pareto (if everyone puts x above y, every voter gives x more points), and is not a dictatorship. By Arrow's theorem it must violate IIA. The example profile shows this violation clearly.

Example 7.16

1. **With four candidates:** Borda B = 22 > D = 21. Society puts B ahead of D.
2. **C withdraws from the race.** C was neither winning nor second; C is an "irrelevant" candidate with respect to the B–D ranking. C is deleted from the rankings: 4 × A ≻ D ≻ B, 4 × B ≻ D ≻ A, 3 × D ≻ B ≻ A, 2 × A ≻ B ≻ D.
3. **Borda (2, 1, 0) for three candidates:**
   - A: $4 \times 2 + 0 + 0 + 2 \times 2 = 12$.
   - B: $4 \times 0 + 4 \times 2 + 3 \times 1 + 2 \times 1 = 13$.
   - D: $4 \times 1 + 4 \times 1 + 3 \times 2 + 2 \times 0 = 14$.
4. **Result:** D = 14 > B = 13. The social order has flipped.

No voter's B–D order has changed: 6 voters prefer B to D and 7 prefer D to B, both before and after. But the social ranking has flipped from B ≻ D to D ≻ B. Borda violates IIA.

candidate	4 candidates	without C
A	18	12
B	**22**	13
C	17	—
D	21	**14**

Table 7.15: Borda scores when C withdraws (4 candidates: 3, 2, 1, 0; 3 candidates: 2, 1, 0).

Why does this happen? By the interpretation in Section 7.2.3, a candidate's Borda score is the number of candidates the voters put below it. When C withdraws, each candidate loses 1 point from every voter who put C below it. The voters who put C below B are G1, G3, and G4: $4 + 3 + 2 = 9$ people; B falls to $22 - 9 = 13$. Those who put C below D are only G1 and G3: 7 people; D falls to $21 - 7 = 14$. Those who put C below A are G1 and G4: 6 people; A falls to $18 - 6 = 12$. C was giving B more "intermediate points" than it gave D; when C withdraws this difference disappears. The Borda score measures the *distance* between two candidates (how many candidates lie between them), and this distance depends on the other candidates. IIA, however, asks that only the relative order of the two candidates be looked at.

IIA can also be violated without changing the candidate set. Let the 4-voter group $A \succ D \succ B \succ C$ only swap C and B and say $A \succ D \succ C \succ B$. These voters' B–D order does not change (D is still ahead of B). But B's score drops from 1 to 0 for each of them: B falls from 22 to $22 - 4 = 18$. D stays at 21, while C rises to $17 + 4 = 21$. The social B–D order flips again. Although no voter's B–D order has changed, the social order of B and D has changed.

7.5.3 The practical meaning of Arrow's theorem

Arrow's theorem is often misinterpreted. What it says and what it does not say should be separated.

- **What it says:** collective decision rules cannot satisfy all reasonable criteria at once; every rule gives up one condition.
- **What it does not say:** it does not say that voting is meaningless. It does not ban or recommend any particular rule. Nor does it deny that most rules give reasonable results in most profiles; it says only that a condition will fail in *some* profiles.
- **Who gives up what?** Borda gives up IIA (Section 7.5.2). The Condorcet or majority ranking gives up transitivity, and hence the "transitive outcome" part of unrestricted domain (the cycle). A dictatorship gives up fairness.
- **For institutions:** which criterion is given priority should be stated explicitly and documented. Choosing a rule is choosing which flaw to accept.

The ways out of the theorem are to relax the conditions. The first is to restrict the domain of preferences: if not all profiles but only profiles of a certain structure are allowed, positive results are possible. The most important example is single-peaked preferences (Section 7.5.6). The second is to

use intensity information instead of rankings. In Arrow's framework the voter reports only a ranking; there is no difference between "I slightly prefer A to B" and "I strongly prefer A to B." Tools that carry intensity information, such as score voting or money, are outside this framework.

7.5.4 The Gibbard–Satterthwaite theorem

Arrow's theorem is about social rankings and does not deal directly with strategic behavior. The question of strategic voting was answered by Allan Gibbard (1973) and Mark Satterthwaite (1975) independently of each other. Two conditions are needed to state the theorem.

Definition 7.16

For a social choice function (SCF) f:

- **Surjectivity** (onto): every candidate wins in at least one profile.
- **Strategy-proofness**: for every voter i, every profile, and every false ballot $\succ'_i$ of i, the outcome $f(\succ_i, \succ_{-i})$ is not worse than $f(\succ'_i, \succ_{-i})$ according to i's true preference $\succ_i$. Here $\succ_{-i}$ denotes the ballots of the other voters.

Theorem 7.2 (Gibbard 1973, Satterthwaite 1975)

If there are at least 3 candidates and each voter can report any strict ranking over the candidates (unrestricted domain), every social choice function that selects a single winner in every profile, is onto, and is strategy-proof is a dictatorship.

The definition of strategy-proofness reads in game-theoretic language as follows: the truthful ballot is a *dominant strategy* for every voter. Whatever the other voters report, telling the truth is at least as good for the voter as lying. The theorem says that this strong property, with three or more candidates and unrestricted preferences, is possible only for a dictatorship among single-winner rules in which every candidate can win (onto). (A constant rule that always picks the same candidate is also strategy-proof, but it is not onto.) A dictatorship really is strategy-proof: the dictator's first choice wins, the dictator gains nothing by lying, and the ballots of the other voters do not affect the outcome at all.

The example profile makes the theorem concrete. Group 3's tactical vote for C under plurality (Section 7.4.1) and the burying of B under Borda (Section 7.4.2) are proofs that these two rules are not strategy-proof. According to the theorem, this is not a flaw of these two rules alone: *every* onto rule other than a dictatorship can be manipulated in some profile.

Its relation to Arrow: Arrow's theorem is about rankings, and Gibbard–Satterthwaite is about a single winner. The two theorems are closely linked; the Gibbard–Satterthwaite theorem can be derived from Arrow's theorem. In terms of mechanism design, the meaning is this: in a moneyless setting, there is

no general incentive-compatible rule for choosing among three or more alternatives. When money can be used, the situation changes; payments allow the cost of lying to be adjusted, and strategy-proof mechanisms such as the VCG mechanism become possible.

The goal of design is therefore not to eliminate strategic behavior but to limit it. Three ways of doing this are commonly used: voters knowing little about the other ballots (manipulation requires information about the profile), manipulation being computationally hard (Section 7.6.1), and restricting the domain of preferences (Section 7.5.6).

Common mistake

Drawing from the theorem the conclusion that "every voter always lies." The theorem says only that there is *at least one* profile in which the rule can be manipulated. In most profiles and for most voters, an honest vote may be the best option. Another mistake is to think the theorem also holds for two candidates; with two candidates the majority rule is strategy-proof.

7.5.5 May's theorem: simple majority for two candidates

The theorems of Arrow and of Gibbard–Satterthwaite both require at least three candidates. With two candidates the situation is much better. Kenneth May (1952) asked which rule is "fair" when there are two candidates (x and y) and showed that the only rule satisfying three conditions is simple majority. In **simple majority**, the candidate with more votes wins; if the votes are equal, the result is a tie.

Definition 7.17

For a rule with two candidates:

- **Anonymity**: voters' names do not matter; if the votes are permuted among the voters, the result stays the same.
- **Neutrality**: candidates' names do not matter; if every voter swaps x and y, the result also swaps.
- **Positive responsiveness**: if the result is a tie or x is winning, and a voter changes their vote in favor of x (the others staying the same), then x strictly wins.

Theorem 7.3 (May, 1952)

For two candidates, a rule satisfies anonymity, neutrality, and positive responsiveness if and only if it is simple majority.

One direction of the theorem is easy to see: simple majority satisfies all three conditions. It is anonymous, because the outcome depends only on how many votes each candidate gets; which voter cast which vote does not change the counts. It is neutral, because when every voter swaps x and y, the two candidates' vote counts swap and the winner swaps. It is positively responsive, because in a tie, one voter moving to x puts x one vote ahead; if x is already ahead, it increases the margin. The real strength of the theorem lies in the other direction: there is no other rule satisfying these three conditions. Why each condition is necessary is seen from the rules that violate it.

Condition	Meaning	Violating rule
Anonymity	Voters' names do not matter; if votes are permuted the result is the same	Dictator; weighted voting
Neutrality	Candidates' names do not matter; the two candidates are treated symmetrically	"The status quo does not change without ⅔" (qualified majority)
Positive responsiveness	In a tie, or when x is winning, if a voter moves to x then x strictly wins	"Always a coin flip"; "whoever gets an odd number of votes wins"

Table 7.16: The conditions of May's theorem and the rules that violate them.

Example 7.17

In a cooperative, 9 members vote on replacing the current bylaws (the status quo, x) with new bylaws (y). The rule: "the bylaws change only with at least ⅔ of the votes." ⅔ of 9 is 6.

- 5 members say y and 4 say x: y does not reach 6; x stays.
- Now let every member reverse their vote: 5 members say x and 4 say y. x stays again.

Neutrality would require that when the votes are reversed, the outcome also be reversed (y wins). The rule violates neutrality because it favors the status quo. The rule is anonymous (who votes does not matter).

Qualified majority rules are used deliberately for important decisions such as constitutional amendments; the aim is to make the status quo hard to change. May's theorem says this has a price: such a rule does not treat the two options equally. The "always a coin flip" rule does not respond to the votes at all; in the "whoever gets an odd number of votes wins" rule, one voter moving to x can make x lose by turning its vote count from even to odd or from odd to even.

The message of May's theorem is this: with two candidates the debate over rules ends; simple majority is by far the most natural choice. The difficulties (the cycle, Arrow, Gibbard–Satterthwaite) begin with three or more candidates.

7.5.6 An escape route: single-peaked preferences

The common assumption of the Arrow and Gibbard–Satterthwaite theorems is unrestricted domain: every possible preference profile is allowed. But in many real situations preferences are not random; they have a certain structure. The most important of these structures is single-peakedness.

Definition 7.18

Let the candidates be ordered from left to right on an axis (for example the political left–right, an amount of budget, or a location on a street). If a voter has a most-liked position (the **peak**) and the voter's utility falls in whichever direction a candidate moves away from this peak, the voter's preference is **single-peaked**. If all voters' preferences are single-peaked with respect to the same axis, the profile is single-peaked.

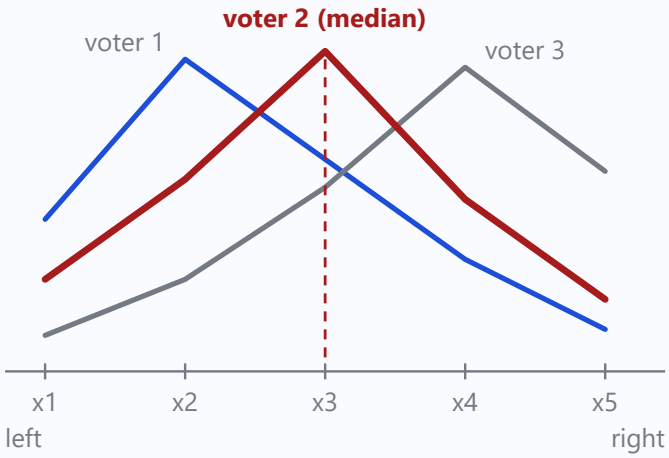


Figure 7.11: Positions x1–x5 on the left–right axis and the single-peaked utility curves of three voters. Voter 1's peak is x2, voter 2's is x3, and voter 3's is x4. The median peak is x3 (voter 2).

In Figure 7.11 each curve has a single summit and falls off on both sides of it. For example, suppose a city council votes on the budget to allocate to a park. Each member has an ideal amount in mind; proposals much higher or much lower please them less. Such preferences are naturally single-peaked.

Theorem 7.4 (Median voter theorem; Black, 1948)

If preferences are single-peaked and the number of voters is odd, then when the peaks are sorted, the peak of the middle voter (the median voter) is the Condorcet winner. Moreover, no Condorcet cycle arises: since single-peakedness is preserved on every subset of candidates, every subset has a Condorcet winner; a subset consisting of the candidates in a cycle could not.

Proof. Let the number of voters be $n = 2k + 1$ and the median peak t. By definition there are at least $k + 1$ voters whose peak is at or to the left of t, and at least $k + 1$ voters whose peak is at or to the right of t. Consider a candidate z to the right of t. For a voter whose peak is at or to the left of t, both t and z are to the right of the peak, and t is closer to the peak. By single-peakedness, this voter prefers t to z. These voters number at least $k + 1$, that is, they are a majority; t beats z. For a candidate to the left of t, the same argument is made with the voters whose peak is at or to the right of t. So t beats every rival and is the Condorcet winner. □

The theorem has two important consequences. First, there are no Condorcet cycles in single-peaked profiles; this explains why the fact that real preferences are often organized along an axis makes cycles rare. Second, the rule "report your peak, pick the median" is strategy-proof. By reporting a false peak, a voter can only *move the median away* from their own peak. A voter whose peak is to the left of the median does not change the median by reporting a peak further left; by reporting a peak to the right, beyond the median, they shift the median to the right, that is, away from themselves. There is no lie that would bring the median closer to their own peak.

Example 7.18

Let the peaks of five voters on the axis be 2, 3, 7, 8, 9. The value in the middle (third) of the sorted list is 7; the median peak is 7.

- Comparing 7 with a position to its left (for example 5), the 3 voters with peaks 7, 8, 9 choose 7; 7 wins 3 – 2.
- Comparing 7 with a position to its right (for example 8.5), the 3 voters with peaks 2, 3, 7 choose 7; 7 again wins 3 – 2.

The average of the peaks is $(2 + 3 + 7 + 8 + 9)/5 = 5.8$, but the average is not the Condorcet winner: comparing 5.8 with 7, the 3 voters with peaks 7, 8, 9 choose 7. If the voter with peak 2 reports their peak as 0, the sorted list becomes 0, 3, 7, 8, 9 and the median is still 7; the lie gains nothing.

The median voter theorem does not contradict the Gibbard–Satterthwaite theorem; it removes one of its assumptions, unrestricted domain. The Gibbard–Satterthwaite theorem speaks about all profiles; once the domain is restricted to single-peaked profiles, the theorem no longer applies. The **median mechanism** used for facility location in mechanism design is an application of this idea: if a library or hospital is to be placed on a street and everyone wants the nearest location, the median of the reported locations is chosen and nobody can gain by reporting a false location.

Common mistake

- **Taking the mean instead of the median:** the mean is affected by extreme values and is not strategy-proof; a voter can pull the mean toward themselves by exaggerating their peak.

- **Assuming every preference is single-peaked:** single-peakedness is an assumption. For example, the preference of a voter who says "either a very large budget or no budget; the middle is the worst" is not single-peaked. If there are such voters, the median theorem cannot be applied.

7.5.7 The four results side by side

The four results in this section are compared in Table 7.17.

Theorem	Assumptions	Conclusion	Escape route
Arrow (1951)	≥ 3 candidates, unrestricted domain, Pareto, IIA	The rule is a dictatorship	Restrict the preference domain (single-peaked); give up IIA; cardinal information instead of rankings (scores, intensity)
Gibbard–Satterthwaite (1973, 1975)	≥ 3 candidates, unrestricted domain, onto (every candidate can win), strategy-proof	The rule is a dictatorship	Single-peaked preferences; randomized rules (e.g. random dictator); money (VCG)
May (1952)	2 candidates; anonymity, neutrality, positive responsiveness	The rule is simple majority (and conversely)	Not needed: a characterization, not an impossibility
Single-peaked preferences (Black, 1948)	Candidates on an axis, each voter has a single peak, an odd number of voters	The median voter's peak is the Condorcet winner; the median rule is strategy-proof	Is itself an escape route: restricts the unrestricted domain assumption

Table 7.17: Assumptions, conclusions, and escape routes of the four results.

The "random dictator" rule in the table deserves an explanation: one of the voters is chosen by lot and that voter's first choice wins. This rule is strategy-proof (for the chosen voter, lying only costs them their favorite) and no voter is a dictator in advance. Since the Gibbard–Satterthwaite theorem holds only for deterministic (non-randomized) rules, this rule falls outside the theorem.

In the theorems of Arrow and of Gibbard–Satterthwaite, the dictatorship appears when all the assumptions are demanded together. When one assumption is relaxed, positive results become possible: two candidates (May), single-peaked preferences (Black), randomness, or money. The practical value of these theorems is that they show the designer which doors are closed and which are open.

7.6 Computational social choice and applications

Social choice meets computer science and finds counterparts in real institutions. The computational cost of rules, collective decisions on propositions, the rules used in real elections, and proportional representation, in which several seats are allocated, are the main topics of this meeting.

7.6.1 Computational social choice

Computational social choice is the intersection of computer science, economics, and social choice. In this field, rules are studied not only by which criteria they satisfy but also by their computational cost. Three kinds of questions stand out: how hard is it to compute the winner, how hard is it to compute a manipulation, and how much information must be collected from voters to determine the winner?

Computing the winner. Plurality and Borda are computed in $O(nm)$ time for n voters and m candidates; each ballot is read once (Section 7.2.10). For Copeland, the pairwise comparison table is filled first: for each of the $m(m-1)/2$ pairs, n ballots are scanned; the time is $O(nm^2)$. By contrast, finding the winner under the Kemeny rule and the Dodgson rule is NP-hard (Bartholdi, Tovey, and Trick, 1989). The **Dodgson rule** was proposed by Lewis Carroll (real name Charles Dodgson), the author of *Alice in Wonderland*: the candidate who needs the fewest *adjacent swaps* (exchanging the positions of two neighboring candidates in a ranking) on the ballots to become the Condorcet winner wins. If a Condorcet winner exists, it needs zero changes and wins. The idea is natural, but computing this smallest number is in general very hard.

There is an irony here: some rules that satisfy the Condorcet criteria and decide sensibly in a cycle (Kemeny, Dodgson) are the computationally most expensive rules. In practice with few candidates this is not a problem; with many candidates, approximation algorithms and integer programming solvers are used.

Computing manipulation. The Gibbard–Satterthwaite theorem says that manipulation is possible, but how hard it is to *find* a profitable false ballot is a separate question. A single manipulator finds a profitable ballot under plurality and Borda in polynomial time with a greedy algorithm: place the favorite first and fill in the remaining positions so as to threaten the favorite as little as possible. Under IRV, however, the problem is NP-hard (Bartholdi and Orlin, 1991). This hardness can be regarded as a “barrier” against manipulation. But the barrier holds only for the worst case: although some instances of an NP-hard problem are hard, most instances encountered in real elections can be solved easily.

Preference elicitation. How much information is needed to learn a voter’s full ranking? For m candidates there are $m!$ different rankings. Each yes/no question (one bit) halves the number of possible rankings at best. A full ranking therefore needs at least $\log_2(m!)$ bits. For $m = 10$, $10! = 3{,}628{,}800$ and $\log_2(3{,}628{,}800) \approx 21.8$, that is, about 22 bits. In large elections or online systems with many candidates this communication cost matters. **Preference elicitation** studies asking voters questions step by step and collecting only the information sufficient to determine the winner. For example, under plurality the first choice alone is enough; asking for the full ranking is unnecessary.

The field has two more topics: **fair division** (sharing divisible or indivisible resources without creating envy) and **judgment aggregation** (Section 7.6.2).

7.6.2 Judgment aggregation: the doctrinal paradox

Voting is not used only to choose among candidates. A panel of judges, a board of directors, or a scientific committee often decides on *propositions*: "is the contract valid?", "was the contract breached?". Turning individual judgments into a collective judgment is called **judgment aggregation**.

Consider a court with three judges. The defendant pays damages only if both propositions p (the contract is valid) and q (the contract was breached) are true. The conclusion proposition is $p \wedge q$ ("p and q").

Judge	p	q	Conclusion: p ∧ q
Judge 1	yes	yes	yes
Judge 2	yes	no	no
Judge 3	no	yes	no
Majority	yes (2–1)	yes (2–1)	no (1–2)

Table 7.18: The doctrinal paradox: three judges, two premises, one conclusion.

Each judge is consistent: Judge 1 accepts both premises and awards damages; Judges 2 and 3 each reject one of the premises and oppose damages. But when the propositions are voted on one by one, the majority says "p is true, q is true, $p \wedge q$ is false." This is a logically inconsistent collective judgment. This situation is called the **doctrinal paradox** (discursive dilemma; Kornhauser and Sager, 1986).

The practical consequence of the paradox is that the decision depends on procedure. If the court decides *by premises* (first p and q are voted on separately, and the conclusion is derived from them), the defendant pays. If it decides *by conclusion* ($p \wedge q$ is voted on directly), the defendant does not pay. Two different decisions come from the same judgments. This is the counterpart of the Condorcet cycle in judgments: individuals are consistent, the majority is inconsistent.

List and Pettit (2002) turned this observation into a theorem: there is no aggregation rule that is anonymous, systematic (each proposition is decided by the same method and only according to the votes on that proposition), and always produces a consistent collective judgment. This is the counterpart of Arrow's theorem for judgments.

7.6.3 Voting rules in the real world

Most of the rules in this chapter are used in real institutions:

- **The two-round system:** presidential elections in Türkiye and France. If no majority is reached in the first round, the top two candidates compete in the second round.
- **Instant-runoff:** elections to the Australian House of Representatives and some US cities. The voter gives a full ranking.
- **Plurality:** single-member districts in the United Kingdom and the United States. In each district the candidate with the most votes is elected.

- **Borda-like scoring:** in the Eurovision Song Contest, each country gives 12, 10, 8, 7, … points to the songs it likes; similar scoring is used in sports awards (for example in most valuable player votes).
- **Approval:** some scientific society elections; “times when I am available” polls.
- **Online:** Reddit and forum voting, community polls, class and club decisions.

Eurovision’s score vector (12, 10, 8, 7, 6, 5, 4, 3, 2, 1, 0, …) is a positional scoring rule. Giving first place 2 points more than second place is meant to reward the first choice a little more than Borda does. This design choice is also an example of the discussion in Section 7.2.5: the score vector sets the balance between compromise candidates and candidates with strong base support.

7.6.4 Several seats: proportional representation and D’Hondt

The rules so far choose a single winner. In parliamentary elections, however, several seats, say S, are allocated from one electoral district. The aim then is not to find a winner but to share the seats among parties. According to the principle of **proportional representation**, each party’s share of seats should be close to its share of votes. Since seats are integers, exact proportionality is generally impossible, and how to round is a choice of method.

Definition 7.19

D’Hondt method: each party’s votes are divided by 1, 2, 3, …. All the quotients of all the parties are sorted in a single list from largest to smallest; the largest S quotients each win one seat. A party’s number of seats is the number of its own quotients among these S quotients.

In Türkiye’s parliamentary elections the D’Hondt method is applied in each electoral district; in addition there is a nationwide threshold.

Example 7.19

6 seats; votes A 42,000, B 31,000, C 15,000, D 12,000 (100,000 in total).

1. Each party’s votes are divided by 1, 2, 3, 4 (Table 7.19).
2. All the quotients are arranged from largest to smallest: 42,000 (A ÷ 1), 31,000 (B ÷ 1), 21,000 (A ÷ 2), 15,500 (B ÷ 2), 15,000 (C ÷ 1), 14,000 (A ÷ 3), 12,000 (D ÷ 1), ….
3. The largest six quotients get seats: three of A’s, two of B’s, and one of C’s. D’s 12,000, in seventh place, is left out.

Result: A 3, B 2, C 1, D 0.

Divisor	A	B	C	D
÷1	**42,000**	**31,000**	**15,000**	12,000

Divisor	A	B	C	D
÷2	**21,000**	**15,500**	7,500	6,000
÷3	**14,000**	10,333	5,000	4,000
÷4	10,500	7,750	3,750	3,000
Seats	3	2	1	0

Table 7.19: D'Hondt quotients. The highlighted cells are the largest six quotients.

The division has an interpretation: a party's kth quotient is the number of "votes per seat" if the party takes one more seat when it already has $k - 1$ seats. Each seat is given to the party that would represent the most votes per seat at that moment. As a result, A takes 50% of the seats with 42% of the votes, and D, with 12% of the votes, gets no seats. D'Hondt slightly favors large parties.

Another method is the **largest remainder** (Hare) method. Each party's "entitled" number of seats is computed as vote share × S: A $0.42 \times 6 = 2.52$, B $0.31 \times 6 = 1.86$, C $0.15 \times 6 = 0.90$, D $0.12 \times 6 = 0.72$. Each party first gets as many seats as the integer part of this number: A 2, B 1, C 0, D 0; 3 in total. The remaining 3 seats are given one by one to the parties with the largest fractional parts: C (0.90), B (0.86), D (0.72). A's fractional part (0.52) is the smallest. Result: A 2, B 2, C 1, D 1. With the same votes, D'Hondt gives A 3 seats, and the largest remainder method gives D 1 seat.

As with single-winner rules, every method here is a trade-off. D'Hondt emphasizes government stability, that is, the ability of large parties to form a majority; the largest remainder method emphasizes the representation of small parties.

7.6.5 Which voting rule is fair?

All the results of the chapter show that there is no single answer to this question. Judging fairness depends on criteria such as representation, majority support, protection of minorities, and strategy-proofness, and these criteria conflict with one another.

- **Plurality** is simple and fast, but it can lead to vote splitting and to a winner without majority support (in the example profile, A with 6/13 of the votes).
- **Borda** uses more preference information and promotes the compromise candidate (B), but it is open to strategic ranking, especially burying.
- **Instant-runoff** seeks majority support, but it does not fully solve the problems of monotonicity and strategic voting.
- **Condorcet methods** respect pairwise majorities but need an additional rule in a cycle; those that also decide in a cycle, such as Kemeny, are hard to compute.

The choice of rule should be made according to the context of the decision and the priority criteria. For a club presidency, a fast and understandable plurality may be enough; for choosing a class activity with many candidates, a ranking-based rule (Borda or a Condorcet method) may give a more

balanced outcome. What matters is that the rule is determined explicitly before the vote and that it is known which flaws are accepted.

7.7 Summary

- **Social choice** studies rules that turn individual preferences (the preference profile) into a collective decision (SCF) or a social ranking (SWF). The ballot is the voter's strategy; voting is a moneyless mechanism design problem.
- **Rules:** majority, plurality, the two-round system, Borda, approval, positional scoring, instant-runoff, Condorcet, Copeland, maximin, and Kemeny. In the 13-voter example profile these rules give four different winners: plurality A, two-round and IRV C, Borda and 2-approval B, Condorcet and Kemeny D.
- **The Condorcet paradox:** transitive individual preferences can produce a cyclic social preference ($A \succ B \succ C \succ A$). In a cycle, whoever controls the agenda chooses the outcome; the candidate left for last wins. Kemeny chooses the ranking that disagrees least with the pairwise majorities (in a three-candidate cycle it overrides the weakest majority), but it is NP-hard to compute.
- **Criteria:** plurality can select the Condorcet loser; Borda and IRV may not select the Condorcet winner; IRV is not monotone; Copeland and Kemeny are Condorcet consistent.
- **Strategic voting:** tactical voting under plurality (avoiding vote splitting), burying under Borda. Honesty is not the best strategy under every rule; burying can also backfire.
- **Arrow:** with at least 3 candidates, unrestricted domain, Pareto, IIA, and non-dictatorship cannot all hold together. Borda violates IIA.
- **Gibbard–Satterthwaite:** with at least 3 candidates, every strategy-proof and onto rule is a dictatorship.
- **May and single-peakedness:** with 2 candidates, simple majority is the only rule satisfying anonymity, neutrality, and positive responsiveness. With single-peaked preferences, the median voter's peak is the Condorcet winner and the median rule is strategy-proof.
- **Computational social choice:** plurality and Borda $O(nm)$, Copeland $O(nm^2)$; Kemeny and Dodgson are NP-hard; manipulation under IRV is NP-hard. The doctrinal paradox is the counterpart of the cycle for judgments. D'Hondt slightly favors large parties.

7.8 Key terms

social choice preference profile ballot transitivity social welfare function (SWF) majority rule plurality rule two-round (runoff) system Borda count approval voting positional scoring rule k-approval 2-approval veto rule instant-runoff voting (IRV) Condorcet winner Condorcet loser pairwise majority graph tournament Copeland rule tie-breaking rule Condorcet paradox / cycle sequential pairwise voting agenda (manipulation) Kemeny rule rank aggregation monotonicity

maximin (Simpson) rule strategic voting tactical voting (plurality) vote splitting Duverger's law burying Arrow's impossibility theorem unrestricted domain Pareto (unanimity) condition independence of irrelevant alternatives (IIA) non-dictatorship Gibbard–Satterthwaite theorem surjectivity (onto) May's theorem simple majority anonymity neutrality positive responsiveness qualified majority single-peaked preferences median voter theorem median mechanism computational social choice Dodgson rule preference elicitation judgment aggregation doctrinal paradox proportional representation D'Hondt method largest remainder (Hare) method

7.9 Test Yourself

1. 11 voters: 5 × A ≻ B ≻ C, 4 × B ≻ C ≻ A, 2 × C ≻ B ≻ A. Who are the majority, plurality, IRV, Borda (2, 1, 0), and Condorcet winners? Is there a Condorcet loser in the profile?

   Answer

   First choices are A 5, B 4, C 2. The threshold is 6; since $5 < 6$ there is no majority winner. Plurality selects A. IRV: C (2) is eliminated and the votes pass to the next candidate, B; B $4 + 2 = 6$, A 5; B wins. Borda: A $5 \times 2 = 10$, B $5 \times 1 + 4 \times 2 + 2 \times 1 = 15$, C $4 \times 1 + 2 \times 2 = 8$; B wins. Pairwise: B – A 6 – 5, B – C 9 – 2, C – A 6 – 5. B is the Condorcet winner. A loses 5 – 6 to both rivals: A is the Condorcet loser, and plurality selected exactly A.

2. In the same profile consider the scoring rules $s = (1, a, 0)$ ($0 \le a \le 1$). What is each candidate's score in terms of a? For which values of a does A win? Can C ever win?

   Answer

   A is never second in any ranking: A = 5. B is the first choice of 4 voters and the second choice of $5 + 2 = 7$ voters: B = $4 + 7a$. C is the first choice of 2 voters and the second choice of 4 voters: C = $2 + 4a$. For A to beat B we need $5 > 4 + 7a$, that is, $a < 1/7$ (at $a = 1/7$, A and B are tied). In this range A also beats C ($5 > 2 + 4a$). C cannot win for any a, because $4 + 7a > 2 + 4a$ for every $a \ge 0$.

3. In the same profile the honest plurality winner is A. (a) How can the 2 voters with C ≻ B ≻ A vote under plurality to get a better outcome according to their preferences? (b) If Borda (2, 1, 0) is

used, what happens if the A voters bury the main rival B and report A ≻ C ≻ B?

Answer

(a) C (2 votes) has no chance of winning. If these two voters vote for B, B gets *4 + 2 = 6*, A gets 5, and B wins. According to their true ranking C ≻ B ≻ A, B is better than A: the tactical vote is profitable. (b) For each A voter, B gets 0 points instead of 1 and C gets 1 point instead of 0: B *15 − 5 = 10*, C *8 + 5 = 13*, A 10. C wins. C is the worst candidate for the A voters; the burying has backfired.

4. 7 voters: 3 × A ≻ B ≻ C, 2 × B ≻ C ≻ A, 2 × C ≻ A ≻ B. What are the results of the pairwise comparisons, the Kemeny ranking, and the maximin winner? How should the agenda be set up in sequential pairwise voting for B to win?

Answer

A – B: *3 + 2 = 5* – 2, A ≻ B. B – C: *3 + 2 = 5* – 2, B ≻ C. C – A: *2 + 2 = 4* – 3, C ≻ A. There is a cycle. Agreement scores: A ≻ B ≻ C *5 + 3 + 5 = 13*, B ≻ C ≻ A 11, C ≻ A ≻ B 11, A ≻ C ≻ B 10, B ≻ A ≻ C 10, C ≻ B ≻ A 8. The Kemeny ranking is A ≻ B ≻ C; the weakest majority, C ≻ A (4 − 3), is overridden. Maximin: A's worst comparison is 3 (against C), B's is 2, and C's is 2; A wins. For B to win, B is left for last: first A – C (C wins 4 – 3), then C – B (B wins 5 – 2).

5. Three rules are proposed for decisions with two options (x and y): (i) "The chair's vote counts double, and whoever gets the most votes wins"; (ii) "at least ⅗ of the votes are needed to accept y, otherwise x stays"; (iii) "if the number of votes for x is even, x wins; if odd, y wins." Which condition of May's theorem does each violate?

Answer

(i) Anonymity: the outcome can change when the votes of the chair and another member are swapped. (ii) Neutrality: the two options are not symmetric. For example, with 11 members, if 6 members say y and 5 say x, x stays since $6/11 < 3/5$. When the votes are reversed (6 members say x, 5 say y), x stays again; the outcome does not reverse. (iii) Positive responsiveness: when x is winning, one voter moving to x makes the count odd and makes x lose.

6. The peaks of 7 voters with single-peaked preferences on an axis are 1, 1, 2, 4, 6, 9, 10. Which position is the Condorcet winner? What is the average of the peaks? In the rule "report your peak,

pick the median," what happens if the voter with peak 10 reports their peak as 20 or as 3?

Answer

The median is the fourth value of the sorted list: 4. By the median voter theorem the Condorcet winner is 4. The average is $33/7 \approx 4.71$; comparing 4 with 4.71, the 4 voters with peaks 1, 1, 2, 4 choose 4. If the voter reports 20, the list becomes 1, 1, 2, 4, 6, 9, 20 and the median is still 4; the exaggeration does not work. If they report 3, the list becomes 1, 1, 2, 3, 4, 6, 9 and the median falls to 3; the outcome moves even farther from this voter's peak. Lying does not pay.

7. 7 seats; votes A 47,000, B 33,000, C 20,000. What is the seat allocation under the D'Hondt and largest remainder (Hare) methods?

Answer

D'Hondt quotients: A 47,000, 23,500, 15,667, 11,750; B 33,000, 16,500, 11,000; C 20,000, 10,000. The largest seven quotients are 47,000 (A), 33,000 (B), 23,500 (A), 20,000 (C), 16,500 (B), 15,667 (A), 11,750 (A): A 4, B 2, C 1. Hare: the entitled numbers are A $0.47 \times 7 = 3.29$, B 2.31, C 1.40; the integer parts are 3, 2, 1 (6 in total). The remaining 1 seat goes to the largest fractional part (C, 0.40): A 3, B 2, C 2. D'Hondt gives the large party one seat more.

8. A Kemeny count with 8 candidates is to be done by brute force. How many rankings are tried, and how many pairs are examined in each ranking? At least how many bits are needed to learn one voter's full ranking?

Answer

$8! = 40{,}320$ rankings are tried. Each ranking has $8 \cdot 7 / 2 = 28$ pairs; in total $40{,}320 \times 28 = 1{,}128{,}960$ pair evaluations are needed (each one a scan of n groups). A full ranking needs $\log_2(8!) = \log_2(40{,}320) \approx 15.3$, that is, at least 16 bits (the number of bits must be an integer).

7.10 Further reading

- F. Brandt, V. Conitzer, U. Endriss, J. Lang, A. D. Procaccia (eds.), *Handbook of Computational Social Choice*, Cambridge University Press, 2016. The main reference for computational social choice; its chapters on Kemeny, Dodgson, the complexity of manipulation, and judgment aggregation elaborate the topics of these notes.
- D. Easley, J. Kleinberg, *Networks, Crowds, and Markets*, Cambridge University Press, 2010, Chapter 23 (Voting). Presents a short and readable proof of Arrow's theorem and single-peaked preferences with little mathematics.

- A. Karlin, Y. Peres, *Game Theory, Alive*, AMS, 2017. The chapter on social choice and voting gives proofs of the Arrow and Gibbard–Satterthwaite theorems.
- N. Nisan, T. Roughgarden, É. Tardos, V. Vazirani (eds.), *Algorithmic Game Theory*, Cambridge University Press, 2007, Chapter 9. Places social choice in the framework of mechanism design and contains a proof of the Gibbard–Satterthwaite theorem.

7.11 References

1. K. J. Arrow, *Social Choice and Individual Values*, Wiley, 1951.
2. M. Balinski, H. P. Young, *Fair Representation*, Yale University Press, 1982.
3. J. J. Bartholdi, J. B. Orlin, "Single transferable vote resists strategic voting", *Social Choice and Welfare* 8(4), 1991.
4. J. J. Bartholdi, C. A. Tovey, M. A. Trick, "The computational difficulty of manipulating an election", *Social Choice and Welfare* 6, 1989.
5. J. J. Bartholdi, C. A. Tovey, M. A. Trick, "Voting schemes for which it can be difficult to tell who won the election", *Social Choice and Welfare* 6, 1989.
6. D. Black, "On the rationale of group decision-making", *Journal of Political Economy* 56(1), 1948.
7. F. Brandt, V. Conitzer, U. Endriss, J. Lang, A. D. Procaccia (ed.), *Handbook of Computational Social Choice*, Cambridge University Press, 2016.
8. C. Dwork, R. Kumar, M. Naor, D. Sivakumar, "Rank aggregation methods for the web", *WWW*, 2001.
9. D. Easley, J. Kleinberg, *Networks, Crowds, and Markets*, Cambridge University Press, 2010 (Chapter 23).
10. A. Gibbard, "Manipulation of voting schemes: a general result", *Econometrica* 41(4), 1973.
11. G.-Th. Guilbaud, "Les théories de l'intérêt général et le problème logique de l'agrégation", *Économie appliquée* 5(4), 1952.
12. A. Karlin, Y. Peres, *Game Theory, Alive*, AMS, 2017.
13. J. G. Kemeny, "Mathematics without numbers", *Daedalus* 88(4), 1959.
14. L. A. Kornhauser, L. G. Sager, "Unpacking the court", *Yale Law Journal* 96(1), 1986.
15. C. List, P. Pettit, "Aggregating sets of judgments: an impossibility result", *Economics and Philosophy* 18(1), 2002.
16. K. O. May, "A set of independent necessary and sufficient conditions for simple majority decision", *Econometrica* 20(4), 1952.
17. N. Nisan, T. Roughgarden, É. Tardos, V. Vazirani (eds.), *Algorithmic Game Theory*, Cambridge University Press, 2007 (Chapter 9).
18. M. Regenwetter, B. Grofman, A. A. J. Marley, I. M. Tsetlin, *Behavioral Social Choice*, Cambridge University Press, 2006.
19. M. A. Satterthwaite, "Strategy-proofness and Arrow's conditions", *Journal of Economic Theory* 10, 1975.

PART III

Networks and Efficiency

Chapters 8–9

PART III · CHAPTER 8

Network Design and Routing

How fast a road is depends on how many people choose it. This simple observation turns route choice in traffic and communication networks into a game: every user tries to shorten his or her own travel time, but the choice also changes everyone else's. These notes start from the basic notions of networks and routing and then develop selfish routing, Wardrop equilibrium and the system optimum; Braess's paradox, which shows that a new road can slow everybody down; congestion games and Rosenthal's potential; network formation games, in which the players build the network itself; and the tension between efficiency and fairness in networks.

LEARNING OUTCOMES

Computing a Wardrop equilibrium
Compute the equilibrium of a two-route network by equalizing latencies; find the system optimum with marginal costs and compare it with the equilibrium.

Explaining Braess's paradox
Show numerically why a new road can increase everyone's travel time.

Thinking with potentials
Compute Rosenthal's potential and prove why best-response dynamics must stop.

Modeling network formation
Check in which networks players who form links reach equilibrium.

8.1 Networks and routing

All the ideas in this chapter revolve around a single tension: the users of a network want to shorten their own trips, but each user's choice also lengthens everyone else's. A small game makes this tension concrete. The basic notions of network, link, capacity and latency then form the language of all the calculations that follow.

8.1.1 A first game: Braess's paradox

N drivers want to get from point S to point T. There are two routes: S→A→T (the upper route) and S→B→T (the lower route). Each route has one narrow road and one wide road. The time on a narrow road grows with the number of drivers using it: if *x* drivers are on a narrow road, the trip takes $40 \cdot x / N$ minutes. If all drivers enter the same narrow road, the time rises to 40 minutes; if nobody does, the road is empty. The wide roads, on the other hand, are independent of traffic and always take 45 minutes.

The rule of the game is simple: every driver picks his or her own fastest route. Nobody cares about the others' travel times; everyone looks only at his or her own trip. This is “selfish” behavior, but it is not malicious. A commuter on the way to work also takes the fastest route suggested by a navigation app and is not expected to think about anyone else's trip.

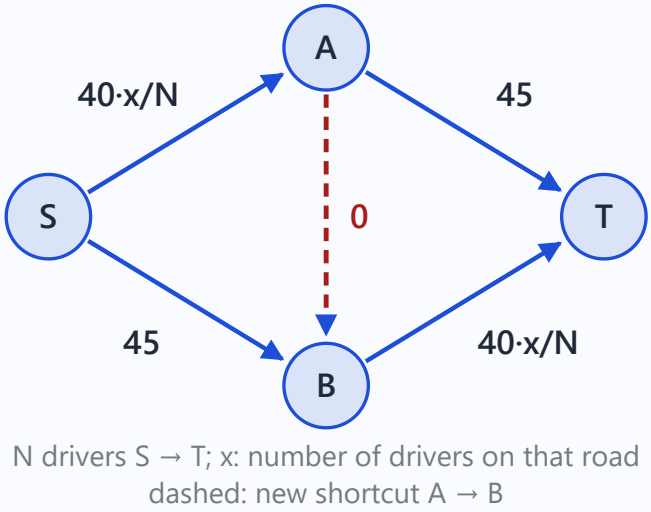


Figure 8.1: The network of the Braess game. S→A and B→T are narrow roads (time $40 \cdot x / N$); A→T and S→B are wide roads (a constant 45 minutes). The dashed red arrow is the new A→B shortcut, which takes 0 minutes.

Figure 8.1 shows the network of the game. Without the shortcut the two routes mirror each other. If the drivers split evenly between the two routes, each narrow road carries $N/2$ drivers and takes $40 \cdot (N/2)/N = 20$ minutes. Each route therefore takes 20 + 45 = 65 minutes. The game asks two questions:

- If a shortcut taking 0 minutes opens from A to B, do trips get shorter?
- When everyone picks his or her own fastest route, is the average time as small as possible?

Common sense answers “yes” to both. A new road adds capacity, and more capacity seems to mean less congestion. If everybody picks the fastest route, the result should surely be “fastest”. The most surprising result of this chapter is that the answer to both questions can be “no”. When the shortcut

opens, every driver switches to the zigzag route (S→A→B→T) and everyone's time rises from 65 minutes to 80 minutes. All the steps of the calculation are given in 8.3.1.

8.1.2 Network design and routing

Decisions about a network are made at two levels. The first level is building the network itself: which points will be connected, how much traffic can each link carry, and how much will construction cost? The second level is using the built network: along which paths will the traffic from one point to another flow?

Definition 8.1

Network design is the problem of determining the nodes, the links, and the capacities and costs of those links. **Routing** is the problem of determining which paths the flow between a source and a destination will use in a given network.

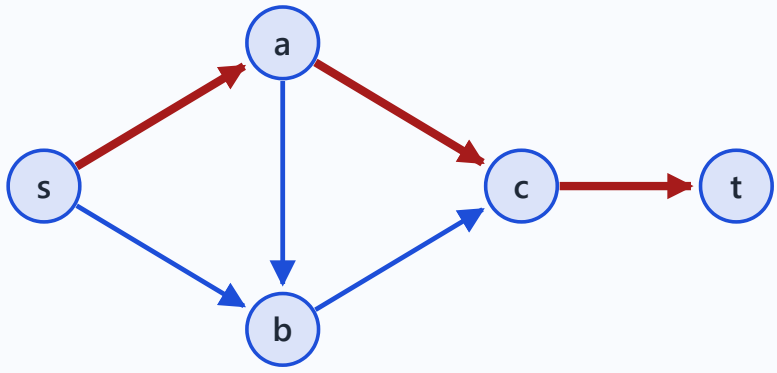


Figure 8.2: The difference between design and routing. Design chooses which edges exist and with what capacity; routing chooses the path the flow takes (red: s→a→c→t) in the network that has been built.

Figure 8.2 shows a small network with five nodes. The designer asks “should there be a link between a and b, and if so, how wide should it be?” Routing asks, once the network is built, “which edges should a packet or vehicle from s to t pass through?” The two problems are linked: a good design must take into account how users will choose paths in that network. Braess's paradox is exactly the price of ignoring this link.

Algorithmic game theory adds a new viewpoint to these problems. In classical network engineering, traffic is something that is “routed”: a central authority decides which path packets or vehicles take. In most real networks, however, the users themselves make the decisions. In a city every driver

chooses his or her own route; on the Internet every network operator directs traffic according to its own interests. Algorithmic game theory studies such traffic and communication networks, in which users try to reduce their own costs. The main application areas are urban traffic, Internet backbones and power grids.

8.1.3 Basic concepts: node, link, capacity, cost

Mathematically, a network is modeled as a **graph**. A graph consists of points and the lines joining them. Four basic concepts are used throughout this chapter.

- **Node**: a point in the network. In urban traffic it is an intersection, on the Internet a **router**, in a power grid a substation.
- **Link** or **edge**: a line joining two nodes. It may be a road, a fiber-optic line or a transmission line. A directed edge (s→a) can be used in one direction only, like a one-way street.
- **Capacity** and **bandwidth**: the flow a link can carry per unit time, such as the number of vehicles a road can pass per hour or the number of bits a line can carry per second. When capacity is exceeded, latency rises rapidly.
- **Cost**: of two kinds. *Construction cost* is paid when the network is built and affects design decisions. *Usage cost* is paid when the network is used: delay, energy spent or fees. Usage cost affects routing decisions.

How the nodes are connected to each other is called the **topology** of the network. Figure 8.3 shows four basic topologies.

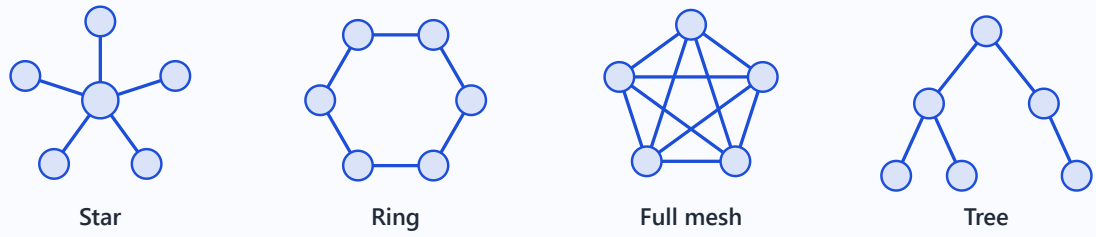


Figure 8.3: Four basic topologies. In a star every node is connected to a hub; in a ring every node has two neighbors; in a full mesh every pair of nodes is directly connected; a tree has no cycles and every node can be reached by a single path.

Each topology has pros and cons. A **star** needs few links (n − 1 links for n nodes) and every node is one step from the hub, but if the hub fails the whole network fails. A **ring** also needs few links (n links) and survives a single break, but distant nodes are many steps apart. A **full mesh** connects every pair directly; distances are shortest, but the number of links is n(n − 1)/2 and grows quickly. A **tree** builds a layered structure and contains no cycles. Trees or layered topologies are common in data centers; a road network in a city is close to a mesh, because there are often several options between two points.

These topologies reappear in 8.5. The question asked there is which topology players who buy the links themselves finally settle on. When links are cheap, a full mesh emerges; when they are expensive, networks close to a star emerge.

8.1.4 Classical routing algorithms

Finding the shortest path in a network where each edge has been assigned a number (a weight) is one of the basic problems of computer engineering. The weight can be a distance, a time or a cost. Table 8.1 summarizes four classical methods used in routing.

Method	What it does	Note
Dijkstra	Single-source shortest paths with nonnegative weights	$O((V + E) \log V)$; the core of OSPF (the link-state protocol that opens the shortest path first)
Bellman–Ford	Allows negative weights; updates by learning from neighbors	$O(V \cdot E)$; its distributed version is RIP (a distance-vector protocol)
Dynamic routing	Recomputes paths when link state and cost change	Adapts to traffic and failures
Load balancing	Spreads traffic over several paths or servers	E.g. ECMP (equal-cost multipath)

Table 8.1: Classical routing algorithms.

In the table, V is the number of nodes and E the number of edges. **Dijkstra's algorithm** starts at the source and, at each step, finalizes “the closest node among those whose distance is not yet final”. For the algorithm to work correctly the weights must be nonnegative: a distance that has been finalized cannot later be shortened by a negative edge. In the **OSPF** (Open Shortest Path First) protocol on the Internet, every router keeps a map of the whole network and runs Dijkstra on that map.

Example 8.1

The network of Figure 8.2 is given the following edge weights: s→a 2, s→b 4, a→b 3, a→c 3, b→c 2, c→t 1. Dijkstra starts at s.

1. Initially the distance of s is 0 and the others are unknown (∞). s is finalized. Its neighbors are updated: 0 + 2 = 2 for a and 0 + 4 = 4 for b.
2. Among the nonfinalized nodes the smallest estimate is a's (2). a is finalized. Its neighbors: 2 + 3 = 5 for b, but b's estimate is already 4; since 5 > 4 it does not change. 2 + 3 = 5 for c.
3. The next smallest is b (4). b is finalized. For its neighbor c, 4 + 2 = 6 > 5; c does not change.
4. Next is c (5). c is finalized. For its neighbor t, 5 + 1 = 6.
5. Finally t is finalized (6). Tracing back, the path is t ← c ← a ← s: the shortest path is s→a→c→t, of length 2 + 3 + 1 = 6. This is the red path in the figure.

The Bellman–Ford algorithm works differently: each node updates its own estimate by looking at the distances its neighbors report, and this continues until the estimates stop changing. At most $V - 1$ rounds suffice and each round scans all the edges; that is why the running time is $O(V \cdot E)$. It is slower than Dijkstra but accepts negative weights. Its most important property is that it can run *in a distributed way*: no node needs to know the whole network, and everyone learns only from neighbors. The **RIP** protocol on the Internet is a distance-vector protocol built on this idea of "learning from the neighbor".

Dynamic routing recomputes paths when a link goes down or its cost changes. **Load balancing** spreads traffic over several paths or servers instead of piling it onto one. **ECMP** (equal-cost multipath) is a simple form of this: if there are several shortest paths of equal length, the traffic is split among them.

All these algorithms share one assumption: the edge weights are *fixed*. Dijkstra accepts that edge a→c takes 3 minutes and treats this number as independent of who uses the edge. In reality, latency depends on how many people choose that path. In the example above, if every driver is told to take s→a→c→t, edge a→c becomes congested and no longer takes 3 minutes. When strategic users can deviate from the recommended route, the equilibrium of the network and the distribution that minimizes total delay come apart. This is what turns routing into a game: each user's "shortest path" depends on the others' choices.

8.1.5 Latency depends on load

To model load-dependent latency, each edge is assigned a function instead of a number.

> **Definition 8.2**
>
> The **latency function** $\ell_e(x)$ of an edge e is the time a user spends on that edge when the flow on it is x. A latency function is *nondecreasing*: as the flow grows, the latency either stays the same or rises; it never falls.

The "nondecreasing" condition rests on common sense: a road does not get faster as more vehicles enter it. Four kinds of latency function are used in this chapter.

- **Constant**: $\ell(x) = b$. A wide highway unaffected by traffic. The 45-minute wide roads of the Braess example are like this.
- **Linear**: $\ell(x) = a \cdot x + b$. Here b is the time on an empty road and a is how much each additional vehicle lengthens the time. The two-route network and the Braess examples use this kind; for instance $\ell(x) = 5 + x/100$ takes 5 minutes on an empty road and every 100 vehicles add one minute.
- **BPR** (US Bureau of Public Roads): $\ell(x) = t_0\,(1 + 0.15\,(x/c)^4)$. Here t_0 is the free-flow time and c is the capacity of the road. It is the standard in traffic planning.

- **Queueing delay**: $\ell(x) = 1/(c - x)$, only for $0 \le x < c$. This is the average delay of a single-server queue (the **M/M/1 queue**). c is the number of packets that can be served per second and x the number arriving per second.

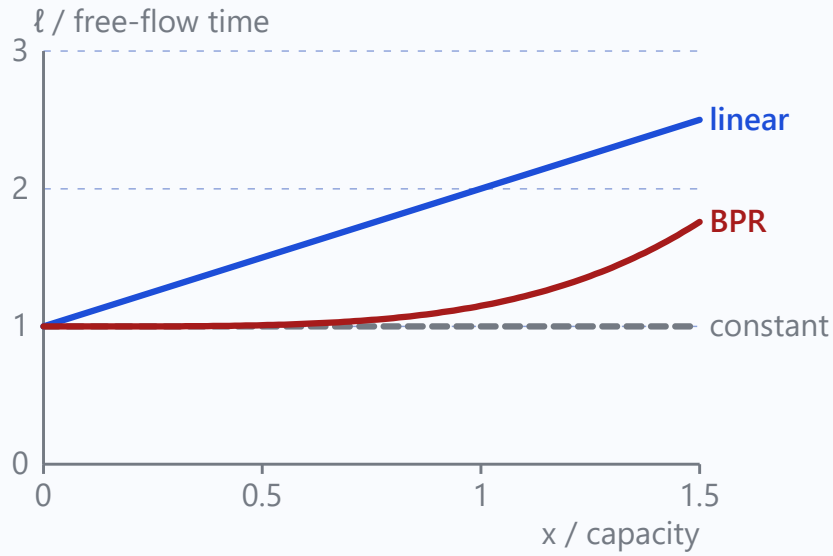


Figure 8.4: Three latency functions. The horizontal axis is the ratio of flow to capacity (x/c), the vertical axis the ratio of latency to free-flow time. Constant latency stays flat; linear latency grows steadily; the BPR curve is almost flat below capacity and rises sharply once capacity is exceeded.

Figure 8.4 compares the shapes of the functions. The behavior of the BPR curve matches everyday experience: when a road is half full traffic is nearly free-flowing, near capacity the slowdown becomes noticeable, and once capacity is exceeded travel time grows rapidly. The fourth power ($(x/c)^4$) produces this "slow at first, then fast" growth.

Example 8.2

Consider a road that takes $t_0 = 10$ minutes when empty and has a capacity of $c = 2{,}000$ vehicles per hour.

1. $x = 1{,}000$: $x/c = 0.5$, $0.5^4 = 0.0625$. The latency is $10 \cdot (1 + 0.15 \cdot 0.0625) \approx 10.09$ minutes; it differs from the empty road by only 6 seconds.
2. $x = 2{,}000$: $x/c = 1$. The latency is $10 \cdot 1.15 = 11.5$ minutes. At capacity the trip is 15% longer.
3. $x = 3{,}000$: $x/c = 1.5$, $1.5^4 = 5.0625$. The latency is $10 \cdot (1 + 0.759) \approx 17.6$ minutes. The flow grew by only 50%, but the increase in latency is more than five times the increase at capacity (7.6 minutes against 1.5 minutes).

The same "wall" appears in queueing delay. Let $c = 100$ packets per second. At $x = 50$ the delay is $1/50 = 0.02$ seconds (20 ms), at $x = 90$ it is $1/10 = 0.1$ seconds (100 ms), and at $x = 99$ it is 1 second.

One property of queueing delay deserves a separate remark: as x approaches c, $1/(c - x)$ grows without bound. When $x \geq c$ the formula loses its meaning: more packets arrive than can be served per second, so the queue grows forever and there is no stable average delay. The formula is therefore valid only for $x < c$.

Common mistakes

- **Treating latency as constant:** The time on a road depends on the number of users. Saying "this road takes 20 minutes" really means "this road takes 20 minutes with this many vehicles".
- **Confusing edge latency with path latency:** ℓ_e is the latency of a single edge. The latency of a path is the sum of the latencies of the edges on it (8.2.1).
- **Using the queueing formula above capacity:** $1/(c - x)$ is undefined or negative for $x \geq c$; this is not a "fast road" but an undefined (ever-growing) queue.

8.2 Selfish routing

In a network with load-dependent latencies two questions stand out: how does traffic distribute itself when users choose on their own, and how good is that distribution in terms of total latency? The answer to the first question is the Wardrop equilibrium; the yardstick for the second is the system optimum.

8.2.1 The model: flow, path latency, total latency

Definition 8.3

Selfish routing is routing in which every user picks the route that minimizes his or her own latency. If there are very many users and each is very small, the flow is taken to be **nonatomic** (divisible): the effect of a single user on the flow is neglected.

The nonatomic assumption is an approximation. A city has thousands of drivers, and one driver entering a road changes its travel time too little to notice. Thinking of traffic as a fluid rather than as individual vehicles makes the calculations easier. When there are few drivers and each one's effect is noticeable (four drivers, say), the model becomes **atomic**; this case is treated in 8.4.

The components of the model are the following:

- r units of traffic go from the source s to the destination t. For example $r = 3{,}000$ cars.
- f_P: the flow on path P, that is, the amount of traffic choosing that path. The flows on all paths add up to r.

- f_e: the total flow on edge e. An edge can be part of several paths; the flows of all the paths containing that edge are added up: $f_e = \Sigma_{P \ni e} f_P$.

With these components two measures are defined. The latency of a path is the sum of the latencies of the edges on it:

$$\ell_P(f) = \Sigma_{e \in P} \ell_e(f_e)$$

The **total latency** of the network is the sum over all edges of flow times latency:

$$C(f) = \Sigma_e f_e \cdot \ell_e(f_e)$$

An example shows why the total latency is written this way. If an edge carries 2,000 cars and each spends 25 minutes on it, the total time spent on that edge is 2,000 × 25 = 50,000 car-minutes. The unit “car-minute” denotes the sum of the minutes spent by every car. Adding these products over all edges gives the total time spent in the network.

The same total can also be computed over paths: multiply the flow of each path by that path's latency and add, $C(f) = \Sigma_P f_P \cdot \ell_P(f)$. The two expressions give the same result. The reason is this: in the product $f_P \cdot \ell_P$, since ℓ_P is a sum of edge latencies, the latency of each edge is multiplied by the flows of all the paths through that edge, and these flows add up to f_e. Dividing the total latency by r gives the average travel time.

Individual route choices can drift away from the distribution that minimizes total latency. Braess's paradox is the most extreme form of this: adding a new link can increase the equilibrium latency. First we need to define how to measure this drift.

8.2.2 Wardrop's principles

The British transport engineer John Glen Wardrop argued in 1952 that traffic flow can distribute itself in two different ways (Wardrop, 1952). These two principles are still the foundation of traffic planning.

Definition 8.4

Wardrop's first principle (user equilibrium). A flow f is a **Wardrop equilibrium** if the latency of every used path is no greater than the latency of any path:

$$f_P > 0 \Rightarrow \ell_P(f) \le \ell_Q(f) \text{ for every path } Q.$$

Wardrop's second principle (system optimum). A flow f^* is a **system optimum** if it minimizes the total latency $C(f)$ over all feasible flows.

The first principle can be read in three equivalent ways:

- All used paths have equal latency, and it is the lowest.
- An unused path is slower (or exactly as fast as the used ones).
- A single user cannot reduce his or her travel time by switching paths.

Why must the latencies of used paths be equal? Suppose both paths are used but Route 1 is faster. Then a driver on Route 2 gains by switching to Route 1. This is incompatible with an equilibrium; an equilibrium is a state in which nobody wants to deviate alone. The third reading shows that the Wardrop equilibrium is in fact the idea of Nash equilibrium adapted to divisible traffic: every user plays a best response to the others' choices.

The second principle is the viewpoint of a planner. A central authority that can manage all traffic by itself chooses the distribution that minimizes the average (total) latency. In this distribution some users may be “sacrificed” to a slower route: their being on the slower route is compensated, in total, by the speed-up of the others. Implementing a system optimum is therefore hard; the sacrificed users must be willing to comply.

In short, the equilibrium explains the individual choices of the users, but it need not coincide with the system optimum. With linear latencies (more generally, with continuous nondecreasing latencies) a Wardrop equilibrium always exists. Moreover, even when there are several equilibria, the equilibrium latency of every edge is the same; whichever equilibrium is found, the travel times the users experience do not change. This result is explained with the Beckmann potential in 8.2.5.

8.2.3 Wardrop equilibrium in a two-route network

The two-route network used throughout the chapter is as follows. 3,000 cars travel from A to B. Route 1 is short but narrow: it takes 5 minutes when empty, but every 100 cars add one minute. Route 2 is long but wide: it takes 20 minutes when empty, and every 200 cars add one minute.

> $$\ell_1(x_1) = 5 + x_1/100,\ \ell_2(x_2) = 20 + x_2/200,\ x_1 + x_2 = 3{,}000$$

Since each route consists of a single edge, path latency and edge latency coincide here. To find the equilibrium we first make two “trials”; these trials show why the equilibrium must lie at a particular point.

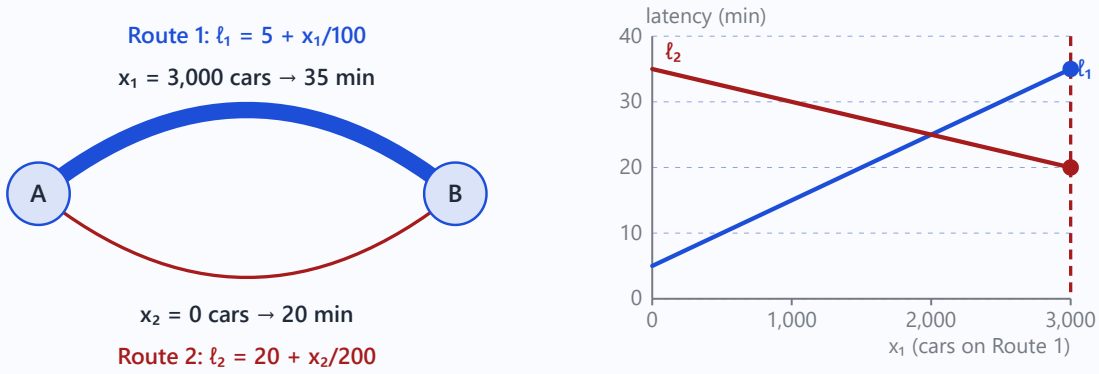


Figure 8.5: Trial 1: all cars on Route 1. On the left the network (a thick line means a lot of flow); on the right the latencies plotted against x_1: the blue line is ℓ_1, the red line is ℓ_2. At $x_1 = 3{,}000$ Route 1 takes 35 minutes and the empty Route 2 takes 20 minutes.

The graph in Figure 8.5 lets us evaluate each trial at a glance. The horizontal axis shows the number of cars on Route 1, x_1; the number on Route 2 is $3{,}000 - x_1$. The blue line rises as x_1 grows (Route 1 gets crowded) and the red line falls as x_1 grows (Route 2 empties). The point where the two lines cross is the point where the two routes take equally long.

Trial	x_1 / x_2	ℓ_1 (min)	ℓ_2 (min)	Who wants to deviate?
Everyone on Route 1	3,000 / 0	35	20	Those on Route 1
Even split	1,500 / 1,500	20	27.5	Those on Route 2
Equilibrium	2,000 / 1,000	25	25	Nobody

Table 8.2: Three trials in the two-route network.

Example 8.3

1. **Trial 1: everyone on Route 1.** $\ell_1(3{,}000) = 5 + 30 = 35$ minutes; the empty Route 2 takes $\ell_2(0) = 20$ minutes. Every driver on Route 1 who switches to Route 2 gains 15 minutes. This is not an equilibrium.
2. **Trial 2: 1,500 / 1,500.** $\ell_1 = 5 + 15 = 20$, $\ell_2 = 20 + 7.5 = 27.5$ minutes. This time those on Route 2 move to Route 1. Again this is not an equilibrium.
3. **The equilibrium condition.** The two trials show that the equilibrium lies somewhere between 1,500 and 3,000. If both routes are used, the latencies must be equal: $5 + x_1/100 = 20 + (3{,}000 - x_1)/200$.
4. **Solution.** To get rid of the fractions, multiply both sides by 200: $1{,}000 + 2x_1 = 4{,}000 + 3{,}000 - x_1$. Collecting the terms in x_1 on one side gives $3x_1 = 6{,}000$, that is, $x_1 = 2{,}000$ and $x_2 = 1{,}000$.
5. **Latencies.** $\ell_1 = 5 + 20 = 25$, $\ell_2 = 20 + 5 = 25$ minutes. The two routes take equally long.

6. **Check.** A single driver who switches routes would also experience about 25 minutes on the other route (because the flow is divisible, the effect of a single driver is neglected). Nobody gains: this is a Wardrop equilibrium. The total latency is *3,000 × 25 = 75,000* car-minutes.

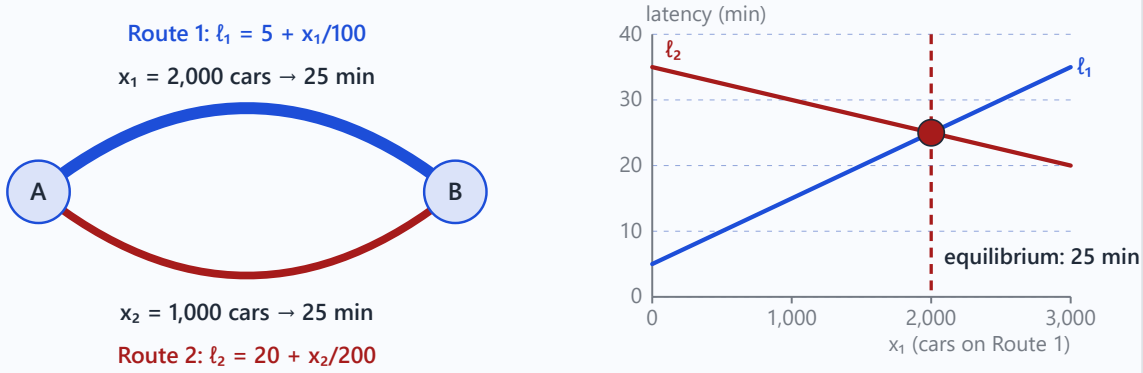


Figure 8.6: The Wardrop equilibrium of the two-route network. 2,000 cars are on Route 1 and 1,000 on Route 2; both routes take 25 minutes. In the graph the equilibrium is the point where the two latency lines cross.

Figure 8.6 shows the equilibrium. To the left of the equilibrium point the blue line lies below the red one: Route 1 is faster and cars move to Route 1, so the point moves right. To its right the blue line is on top: Route 1 is slower and cars move to Route 2, so the point moves left. This "push" from both directions carries the flow to the crossing point. The equilibrium is therefore stable.

The equalization method is valid only when both routes are used. Can a route be faster than the other, empty route even while carrying all the traffic? For instance, if Route 1 were wide enough to stay faster than the empty Route 2 even with 3,000 cars ($\ell_1(3{,}000) \le \ell_2(0)$), the equilibrium would be the state in which everyone is on Route 1. Then the only used route is Route 1 and the unused Route 2 is slower; the second reading of the definition covers exactly this case. In short, if the solution of the equalization equation does not fall between 0 and r, the equilibrium is at an endpoint.

Example 8.4

6,000 cars; Route 1: $\ell_1 = 10 + x_1/200$, Route 2: $\ell_2 = 25 + x_2/100$.

1. First the extreme cases are checked. If everyone is on Route 1, $\ell_1 = 10 + 30 = 40 > \ell_2(0) = 25$; those on Route 1 flee. If everyone is on Route 2, $\ell_2 = 25 + 60 = 85 > \ell_1(0) = 10$; those on Route 2 flee. So both routes are used.
2. Equality: $10 + x_1/200 = 25 + (6{,}000 - x_1)/100$. Multiply both sides by 200: $2{,}000 + x_1 = 5{,}000 + 12{,}000 - 2x_1$.
3. Collecting terms, $3x_1 = 15{,}000$, $x_1 = 5{,}000$, $x_2 = 1{,}000$.

4. Check: $\ell_1 = 10 + 25 = 35$, $\ell_2 = 25 + 10 = 35$ minutes. Both are 35 minutes; the total latency is $6{,}000 \times 35 = 210{,}000$ car-minutes.

8.2.4 Equilibrium is not the system optimum

At the equilibrium everyone travels 25 minutes and the total is 75,000 car-minutes. Is this the smallest possible total latency? To find out, write the total latency as a function of x_1:

$$C(x_1) = x_1 \cdot \ell_1(x_1) + x_2 \cdot \ell_2(x_2),\ x_2 = 3{,}000 - x_1$$

A few values of this function are given in Table 8.3. The table shows that the lowest point of the curve is not at the equilibrium (2,000) but at 1,500.

x_1	0	500	1,000	1,500	2,000	2,500	3,000
C	105,000	86,250	75,000	**71,250**	75,000	86,250	105,000
Φ	82,500	69,375	60,000	54,375	**52,500**	54,375	60,000

Table 8.3: The total latency $C(x_1)$ and the Beckmann potential $\Phi(x_1)$ (car-minutes).

To find the optimum exactly, without scanning a table, we use the notion of **marginal cost**. The marginal cost of a route is the amount that adding one more car to that route adds to the total latency.

Why this amount exceeds the route's own latency can be seen as follows. With x_1 cars on Route 1, add one more car. The new car itself experiences a delay of ℓ_1. But it also makes the road slightly more crowded: the latency rises by $\ell_1' = 1/100$ minute for every car, and all the other x_1 cars on the road experience this increase. In total an extra delay of $x_1 \cdot \ell_1'$ arises. The marginal cost is the sum of these two parts:

$$(x \cdot \ell(x))' = \ell(x) + x \cdot \ell'(x)$$

This formula is the product rule ($(uv)' = u'v + uv'$) applied to this case. Here the derivative can be read as “how much the output grows when the input grows by one unit”. For linear latencies the calculation is easy:

- Route 1: $x_1 \cdot \ell_1 = 5x_1 + x_1^2/100$. The derivative of $5x_1$ is 5; the derivative of $x_1^2/100$ is $2x_1/100 = x_1/50$. The marginal cost is $5 + x_1/50$.
- Route 2: $x_2 \cdot \ell_2 = 20x_2 + x_2^2/200$. The marginal cost is $20 + x_2/100$.

At the system optimum the marginal costs of the two routes must be equal. The reason is the same argument as for the equilibrium, now from the planner's point of view: if the marginal cost of Route 1 were higher, moving one car from Route 1 to Route 2 would reduce the total latency. The planner keeps moving cars until no such improvement remains; where it stops, the marginal costs are equal.

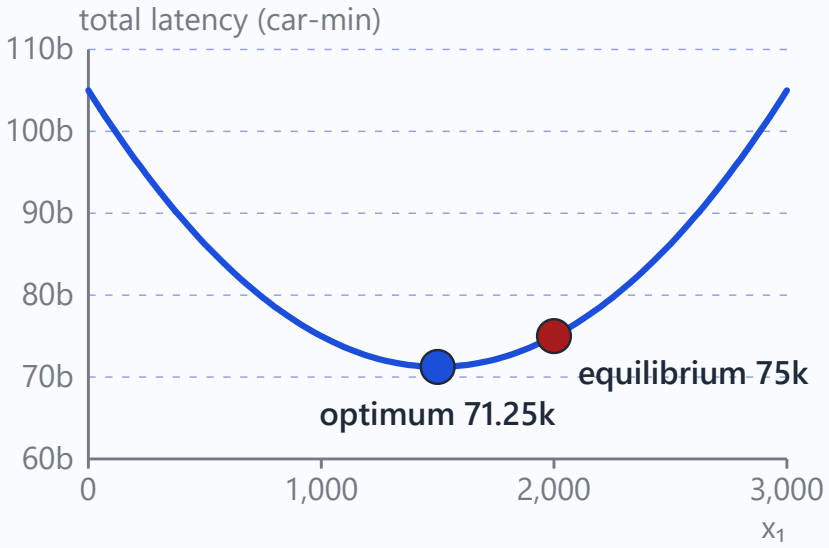


Figure 8.7: The total latency curve $C(x_1)$ (k = thousand car-minutes). The equilibrium ($x_1 = 2{,}000$, 75 thousand) is not at the bottom of the curve; the lowest point is the optimum ($x_1 = 1{,}500$, 71.25 thousand).

Figure 8.7 shows the total latency curve. The curve is bowl-shaped: at the ends (all traffic on one route) the total latency is very large, and somewhere in the middle it is smallest. The equilibrium point lies to the right of the bottom of the bowl.

Example 8.5

1. The marginal costs are equalized: $5 + x_1/50 = 20 + (3{,}000 - x_1)/100$.
2. Multiply both sides by 100: $500 + 2x_1 = 2{,}000 + 3{,}000 - x_1$, that is, $3x_1 = 4{,}500$.
3. $x_1 = x_2 = 1{,}500$. The traffic is split evenly between the two routes.
4. Latencies: $\ell_1 = 20$, $\ell_2 = 27.5$ minutes. Total: $1{,}500 \times 20 + 1{,}500 \times 27.5 = 30{,}000 + 41{,}250 = 71{,}250$ car-minutes.
5. Check: the marginal costs are $5 + 1{,}500/50 = 35$ and $20 + 1{,}500/100 = 35$; they are equal.

	x_1 / x_2	ℓ_1 / ℓ_2 (min)	Total (car-min)
Equilibrium	2,000 / 1,000	25 / 25	75,000
Optimum	1,500 / 1,500	20 / 27.5	71,250

Table 8.4: Equilibrium and optimum in the two-route network.

Table 8.4 compares the two distributions. At the equilibrium the total latency is 3,750 car-minutes above the optimum; as a ratio, $75{,}000 / 71{,}250 \approx 1.053$, about 5% more. This "price" of selfish behavior is studied in the theory under the name **price of anarchy**.

Why does the optimum not arise on its own? At the optimum a driver on Route 1 experiences 20 minutes and a driver on Route 2 experiences 27.5 minutes. Every driver on Route 2 sees that switching to Route 1 would gain 7.5 minutes, and switches. So the optimum is not an equilibrium: traffic left to itself does not stay at the optimum but slides to the equilibrium. To maintain the optimum, the 1,500 drivers on Route 2 would have to accept being "sacrificed".

The source of the inefficiency at equilibrium can also be seen through marginal costs. At the equilibrium ($x_1 = 2{,}000$) the marginal cost of Route 1 is $5 + 2{,}000/50 = 45$ and that of Route 2 is $20 + 1{,}000/100 = 30$. Moving one car from Route 1 to Route 2 would reduce the total latency by about $45 - 30 = 15$ car-minutes. But the driver of that car does not see this: he or she looks only at his or her own time and, experiencing 25 minutes on both routes, is content where he or she is.

Example 8.6

Find the system optimum of the network of Example 8.4 (6,000 cars; $\ell_1 = 10 + x_1/200$, $\ell_2 = 25 + x_2/100$).

1. Marginal costs: for Route 1, $10 + 2x_1/200 = 10 + x_1/100$; for Route 2, $25 + 2x_2/100 = 25 + x_2/50$.
2. Equality: $10 + x_1/100 = 25 + (6{,}000 - x_1)/50$. Multiply both sides by 100: $1{,}000 + x_1 = 2{,}500 + 12{,}000 - 2x_1$, that is, $3x_1 = 13{,}500$.
3. $x_1 = 4{,}500$, $x_2 = 1{,}500$. The latencies are $\ell_1 = 10 + 22.5 = 32.5$, $\ell_2 = 25 + 15 = 40$ minutes.
4. Total: $4{,}500 \times 32.5 + 1{,}500 \times 40 = 146{,}250 + 60{,}000 = 206{,}250$ car-minutes. This is 3,750 less than the 210,000 at equilibrium. At the optimum the drivers on Route 2 are 7.5 minutes slower and want to switch to Route 1.

8.2.5 Two different minimizations: the Beckmann potential

The system optimum is the minimum point of a function (C). Can the Wardrop equilibrium also be written as the minimum point of a function? Martin Beckmann, C. B. McGuire and C. B. Winsten showed in 1956 that it can (Beckmann, McGuire, Winsten, 1956).

Definition 8.5

In a nonatomic routing problem the **Beckmann potential** is

$$\Phi(f) = \sum_e \int_0^{f_e} \ell_e(t)\, dt$$

The integral in the formula is not as hard as it looks. $\int_0^{f_e} \ell_e(t)\, dt$ is the area under the latency curve between 0 and f_e. It is easy to think of it as a sum: let the traffic enter the edge a little at a time. The first driver sees the empty road, that is $\ell_e(0)$; the next ones see a slightly more crowded road; the last

one sees $\ell_e(f_e)$. The area is the sum of the latencies each driver sees *at the moment of entering*. The total latency $f_e \cdot \ell_e(f_e)$, by contrast, charges everyone the latency of the final state; this is why the potential is generally smaller than the total latency.

For a linear latency $\ell(t) = b + a \cdot t$ the area is a trapezoid and is easy to compute: $\int_0^x (b + a\,t)\,dt = b\,x + a\,x^2/2$. In the two-route network:

$$\Phi(x_1) = 5x_1 + x_1^2/200 + 20x_2 + x_2^2/400$$

Take the derivative of this function with respect to x_1. Since $x_2 = 3{,}000 - x_1$, when x_1 grows by one unit x_2 shrinks by one unit. The derivative of the Route 1 part is $5 + x_1/100 = \ell_1$; the Route 2 part decreases by ℓ_2. The result:

$$\Phi'(x_1) = \ell_1(x_1) - \ell_2(x_2)$$

The point where the derivative is zero is the point where $\ell_1 = \ell_2$; this is exactly the Wardrop condition. The derivative of the potential is the latency seen by "the last driver". A selfish driver looks at exactly this: on entering a route, he or she sees the current latency.

Theorem 8.1 (Beckmann, McGuire, Winsten)

If the latency functions are continuous and nondecreasing, a flow f is a Wardrop equilibrium if and only if it minimizes the Beckmann potential Φ over all feasible flows. The system optimum, on the other hand, is the flow that minimizes the total latency $C(f) = \Sigma_e f_e \cdot \ell_e(f_e)$.

The theorem has two important consequences. The first is *existence*: the set of feasible flows is closed and bounded, and a continuous function attains its minimum on such a set; hence a Wardrop equilibrium always exists. The second is *uniqueness*: since the latencies are nondecreasing, Φ is a "bowl-shaped" (convex) function. A bowl-shaped function can have several minimum points, but on the line segment between two minimum points Φ stays constant. This is possible only if the latency of every edge whose flow changes is constant along that segment; so at all minimum points the edge latencies are the same. This is the basis of the sentence in 8.2.2 that "whichever equilibrium is found, the edge latencies are the same".

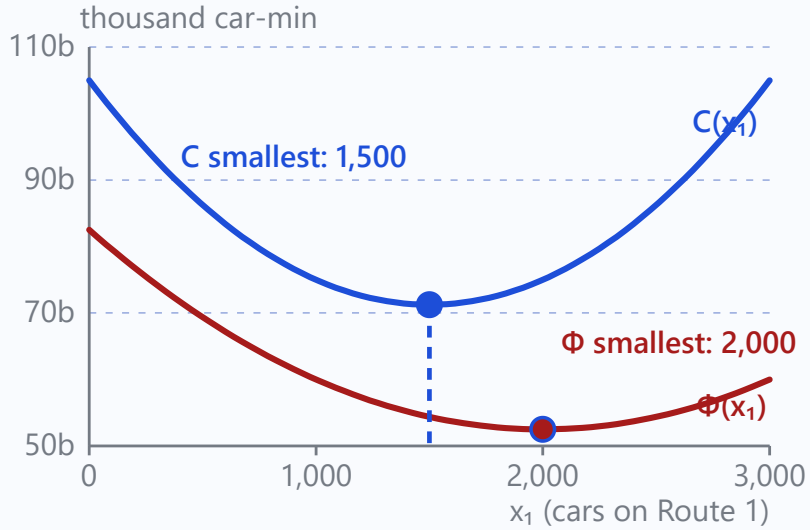


Figure 8.8: Two different minimizations. The blue curve is the total latency $C(x_1)$, the red curve the Beckmann potential $\Phi(x_1)$. The bottom of C is at $x_1 = 1{,}500$ (optimum, 71,250); that of Φ is at $x_1 = 2{,}000$ (equilibrium, 52,500).

Figure 8.8 puts the two curves side by side. The numbers in Table 8.3 say the same thing: Φ takes its smallest value at 2,000 (52,500) and C takes its smallest value at 1,500 (71,250). The equilibrium is the bottom of the potential; the optimum is the bottom of the total latency. The two bottoms are in different places because the derivatives of the two functions differ:

- The derivative of Φ with respect to an edge is ℓ_e: the latency seen by the last driver to arrive.
- The derivative of C with respect to an edge is $\ell_e + f_e \cdot \ell_e'$: the marginal cost.

The difference comes from a single term: $f_e \cdot \ell_e'$. This term is the extra delay a driver imposes on the others merely by being there. The driver does not take this **externality** into account. At the equilibrium on Route 1 this term is $2{,}000 \times 1/100 = 20$ car-minutes: every new driver entering Route 1 slows each of the 2,000 people there by 1/100 minute. The notion of externality is studied separately in 8.5.1.

8.2.6 Code trace: find_eq()

In the two-route network it is easy to find the equilibrium by solving an equation. When the latencies are not linear (BPR, for example), solving the equation by hand becomes hard, and a numerical method is used instead. The function below searches for the equilibrium by **bisection**. Its inputs are the two latency functions `l1` and `l2`, the total traffic `r` and the desired precision `eps`.

```
def find_eq(l1, l2, r, eps=50):
  if l1(0) >= l2(r): return 0   # Route 1 unused
  if l1(r) <= l2(0): return r   # all on Route 1
  lo, hi = 0, r
  while hi - lo > eps:
    x = (lo + hi) / 2
    if l1(x) < l2(r - x):
      lo = x      # Route 1 faster: raise
    else:
      hi = x      # Route 1 slower: lower
  return (lo + hi) / 2
```

The code works as follows:

- The first two lines check the extreme equilibria. If Route 1, even when empty, is no faster than a completely full Route 2 (`l1(0) >= l2(r)`), nobody uses Route 1 and the equilibrium is 0. If Route 1, even when completely full, is no slower than an empty Route 2, everyone uses Route 1 and the equilibrium is r.
- `lo` and `hi` are the ends of the interval known to contain the equilibrium. At first the interval is *[0, r]*: the number of cars on Route 1 must be between 0 and r.
- The loop runs as long as the width of the interval is greater than `eps` .
- In each round the midpoint `x` of the interval is tried. There are `x` cars on Route 1 and `r - x` on Route 2.
- If `l1(x) < l2(r - x)` , Route 1 is still faster. Cars want to move to Route 1; the equilibrium lies further right. The left end of the interval moves to `x` .
- Otherwise Route 1 is too crowded; the equilibrium lies further left. The right end of the interval moves to `x` .
- When the loop ends, the middle of the interval is returned.

The reason the method works is this: the difference $g(x) = \ell_1(x) - \ell_2(r - x)$ never decreases with x, because as x grows Route 1 gets slower and Route 2 gets faster. If the latencies are strictly increasing, g is strictly increasing too; a strictly increasing function has at most one zero, and that zero is the equilibrium. If Route 1 is faster at $x = 0$ ($g < 0$) and slower at $x = r$ ($g > 0$), as in the example, the zero lies between the two. If one route is faster in every case (for example $g(r) \le 0$), the equilibrium is at an endpoint and everyone uses that route; the first two lines of the code catch this case beforehand and return the endpoint exactly. Since each round halves the interval and always keeps the zero inside it, the search closes in on the zero.

Round	`x`	$\ell_1(x)$	$\ell_2(r-x)$	Decision	New interval	Width
—	—	—	—	start	[0, 3,000]	3,000
1	1,500	20	27.5	`lo = x`	[1,500, 3,000]	1,500
2	2,250	27.5	23.75	`hi = x`	[1,500, 2,250]	750

Round	x	$\ell_1(x)$	$\ell_2(r-x)$	Decision	New interval	Width
3	1,875	23.75	25.63	`lo = x`	[1,875, 2,250]	375
4	2,062.5	25.63	24.69	`hi = x`	[1,875, 2,062.5]	187.5
5	1,968.75	24.69	25.16	`lo = x`	[1,968.75, 2,062.5]	93.75
6	2,015.63	25.16	24.92	`hi = x`	[1,968.75, 2,015.63]	46.88

Table 8.5: Trace of `find_eq(l1, l2, 3000)` ($\ell_1 = 5 + x/100$, $\ell_2 = 20 + x/200$, `eps = 50`).

After the sixth round (Figure 8.9) the width is 46.88, and since $46.88 \le 50$ the loop stops. The function returns $(1{,}968.75 + 2{,}015.63)/2 \approx 1{,}992.19$. The true equilibrium is 2,000; the error is about 8 cars, less than half of `eps`.

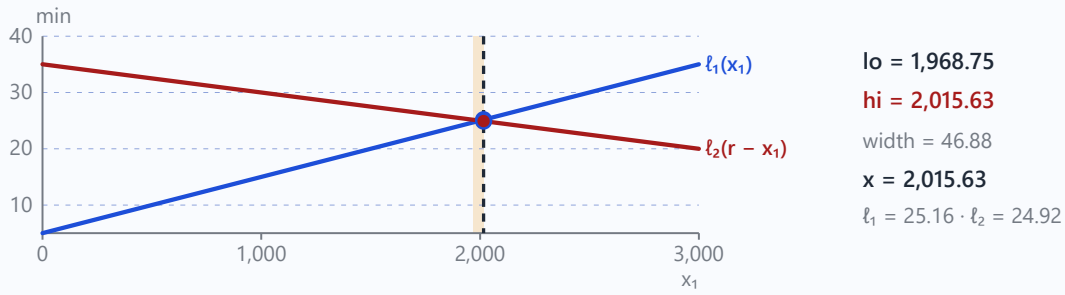


Figure 8.9: The sixth round of `find_eq`. The blue line is $\ell_1(x_1)$, the red line $\ell_2(r - x_1)$; the orange band is the remaining interval [1,968.75, 2,015.63], and the dashed line is the tried value $x = 2{,}015.63$. The interval always contains the crossing point (2,000).

Since each round halves the interval, after k rounds its width is $r/2^k$, and the error is at most that. With 3,000 cars the width falls to $3{,}000/64 \approx 46.9$ in 6 rounds. For a more precise result each extra round halves the error; 10 more rounds reduce it about a thousandfold. This is why the method is very fast.

Example 8.7

Trace the first two rounds of `find_eq` on the network of Example 8.4 ($r = 6{,}000$; $\ell_1 = 10 + x/200$, $\ell_2 = 25 + x/100$).

1. The initial interval is [0, 6,000]. In round 1, $x = 3{,}000$: $\ell_1 = 10 + 15 = 25$, $\ell_2(3{,}000) = 25 + 30 = 55$. $25 < 55$, so Route 1 is faster: `lo = 3000`, interval [3,000, 6,000].
2. In round 2, $x = 4{,}500$: $\ell_1 = 10 + 22.5 = 32.5$, $\ell_2(1{,}500) = 25 + 15 = 40$. $32.5 < 40$: `lo = 4500`, interval [4,500, 6,000].
3. The equilibrium (5,000) lies inside this interval. The search stops in the seventh round and returns about 4,992.

This method works only in two-route networks, because there is a single unknown (x_1). In general networks with many routes, the equilibrium is the flow that minimizes the Beckmann potential and is found with **convex optimization** methods. The method most widely used in traffic planning is the Frank–Wolfe algorithm: at each step it finds shortest paths under the current latencies (with Dijkstra, for example) and shifts the flow a little toward those paths.

8.2.7 Common mistakes in selfish routing

Common mistakes

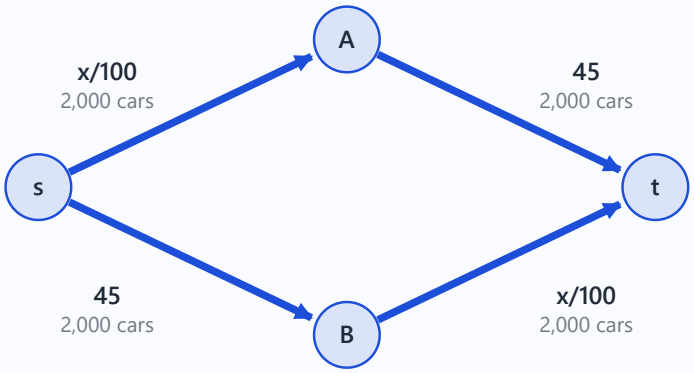
- **Taking the equilibrium to be an "even split":** The equilibrium does not split traffic evenly among the routes; it equalizes the latencies. In the two-route network the even split (1,500 / 1,500) is not the equilibrium but the optimum.
- **Using the equalization equation blindly:** The equation is valid only when both routes are used. If the solution comes out negative or greater than r, the equilibrium is at an endpoint.
- **Equalizing latencies at the optimum:** At the optimum what is equalized is the *marginal costs* ($\ell + x \cdot \ell'$), not the latencies. At the optimum the route times are usually different (20 and 27.5).
- **Doubling the constant when differentiating:** $(x \cdot (5 + x/100))' = 5 + x/50$, not $10 + x/50$. The extra term comes only from $x \cdot \ell'$.
- **Mistaking total latency for an average:** 75,000 car-minutes is a total; the average travel time is $75{,}000 / 3{,}000 = 25$ minutes.
- **Confusing the potential with the total latency:** Φ is not a "cost" but an auxiliary function used to find the equilibrium. Its value (52,500) does not represent an actual total time.

8.3 Braess's paradox

In 1968 the German mathematician Dietrich Braess published a short paper on traffic planning (Braess, 1968). What it showed was simple but disturbing: adding a new road to a network can lengthen everyone's trip, because of selfish route choice.

8.3.1 Braess's paradox step by step

The setup is the same as the game in 8.1.1, except that the number of drivers is now concretely 4,000. 4,000 cars go from s to t. s→A and B→t are narrow roads: with x cars they take $x/100$ minutes. A→t and s→B are wide highways: they take 45 minutes regardless of traffic. Putting $N = 4{,}000$ into the formula $40 \cdot x/N$ of the game gives $x/100$; the two descriptions are the same.



Route	cars	time
s→A→t	2,000	65 min
s→B→t	2,000	65 min
s→A→B→t	—	—

Average: 65 min

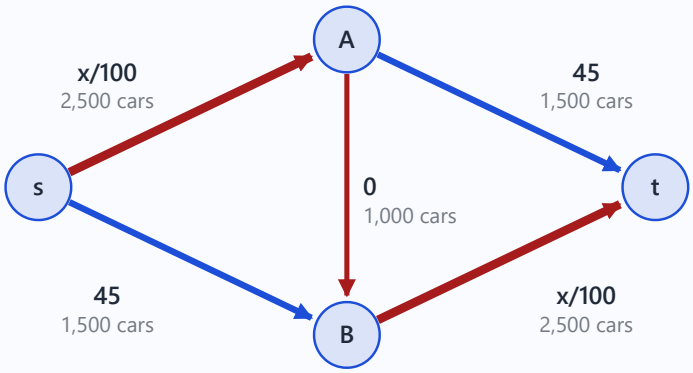
Total: 260,000 car-min

Figure 8.10: The equilibrium of the Braess network without the shortcut. The cars split 2,000 / 2,000 between the two routes; each route takes *2,000/100 + 45 = 65* minutes. The total latency is 260,000 car-minutes.

Example 8.8

1. **Without the shortcut.** There are two routes: upper (s→A→t) and lower (s→B→t). The network is symmetric; each route has one narrow and one wide road. If there are x cars on the upper route, the upper route takes $x/100 + 45$ and the lower route $(4{,}000 - x)/100 + 45$ minutes. Equality gives $x = 2{,}000$. Each route takes $20 + 45 = 65$ minutes. A driver who switches routes gains nothing: this is a Wardrop equilibrium. By symmetry it is also the optimum.
2. **The shortcut opens.** The city opens a very short connecting road from A to B that takes 0 minutes. At first nobody uses it.
3. **The first driver tries it.** A single driver tries the zigzag route (s→A→B→t). There are 2,000 cars on s→A (20 minutes), A→B takes 0 minutes, and there are 2,000 cars on B→t (20 minutes). The total is $20 + 0 + 20 = 40$ minutes. On the old route this driver experienced 65 minutes; he or she gains 25 minutes. Individually this is very profitable, and the driver switches.
4. **The shift continues.** Say 1,000 drivers have moved to the zigzag; 1,500 drivers remain on each old route. Each narrow road now carries $1{,}500 + 1{,}000 = 2{,}500$ cars and takes 25 minutes. The old routes take $25 + 45 = 70$ and the zigzag takes $25 + 0 + 25 = 50$ minutes. The zigzag is still much faster; the shift continues.
5. **Everyone on the zigzag.** All 4,000 drivers are on the zigzag. Each narrow road carries 4,000 cars (40 minutes). The zigzag takes $40 + 0 + 40 = 80$ minutes.
6. **Equilibrium check.** A driver who goes back to an old route uses one narrow road (40 minutes) and one wide road (45 minutes): $40 + 45 = 85$ minutes. Since 85 > 80, nobody goes back. This is the new Wardrop equilibrium.

Result: a road was added and everyone lost 15 minutes. The total latency rose from 260,000 to *4,000 × 80 = 320,000* car-minutes.

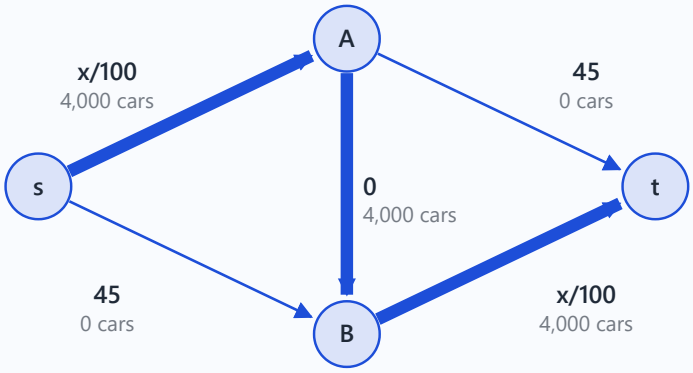


Route	cars	time
s→A→t	1,500	70 min
s→B→t	1,500	70 min
s→A→B→t	1,000	50 min

The zigzag is still the fastest

50 min < 70 min

Figure 8.11: During the shift: 1,000 drivers on the zigzag. The narrow roads carry 2,500 cars (25 minutes); the old routes take 70 and the zigzag 50 minutes. The zigzag is still the fastest route.



Route	cars	time
s→A→t	0	85 min
s→B→t	0	85 min
s→A→B→t	4,000	80 min

65 → 80 min

260k → 320k car-min

Figure 8.12: The equilibrium of the network with the shortcut: all 4,000 drivers are on the zigzag, and everyone takes 80 minutes. The old routes are empty because they would now take 85 minutes. The average time has risen from 65 to 80 minutes, and the total latency from 260 thousand to 320 thousand car-minutes.

Figure 8.10, Figure 8.11 and Figure 8.12 show this process at three moments. Table 8.6 summarizes the same process in numbers.

z	Cars on a narrow road	Narrow road (min)	Old route (min)	Zigzag (min)
0	2,000	20	65	40
1,000	2,500	25	70	50
2,000	3,000	30	75	60
3,000	3,500	35	80	70

z	Cars on a narrow road	Narrow road (min)	Old route (min)	Zigzag (min)
4,000	4,000	40	85	80

Table 8.6: The shift after the shortcut opens (z: number of cars on the zigzag; the remaining traffic splits evenly between the two old routes).

In every row of the table the zigzag is faster than the old routes (the gap shrinks from 25 minutes to 5 minutes). But as z grows both times rise: every driver who switches slows down both those on the zigzag and those on the old routes. At each step the drivers move to the "faster" route, but because they all move together, in the end everyone is slower than at the start. Added capacity combined with selfish choice has led to a worse equilibrium.

Example 8.9

In the same network (narrow roads $x/100$, wide roads 45 minutes) let the number of cars be 6,000. Find the equilibrium time before and after the new road.

1. **Before:** by symmetry 3,000 / 3,000. Each route takes $30 + 45 = 75$ minutes.
2. **After: can everyone be on the zigzag?** With 6,000 cars on the narrow roads the zigzag takes $60 + 0 + 60 = 120$ minutes. A driver who goes back to an old route experiences $60 + 45 = 105$ minutes. Since 105 < 120, going back pays; this is not an equilibrium.
3. **The equilibrium in which traffic splits.** Let z cars be on the zigzag and y on each old route. Both the upper route and the zigzag use the road s→A; the flow on it is $y + z$. The same holds for B→t. The three used routes must take equal time: the zigzag takes $2(y + z)/100$ and an old route $(y + z)/100 + 45$. Equality gives $(y + z)/100 = 45$, that is, $y + z = 4{,}500$.
4. **Flow conservation.** All the cars must be counted: $2y + z = 6{,}000$. The two equations give $y = 1{,}500$ and $z = 3{,}000$.
5. **Time.** 4,500 cars on the narrow roads (45 minutes). Every route takes $45 + 45 = 90$ minutes. The paradox appears again: from 75 to 90. This time some drivers stay on the old routes, but everyone's time is 15 minutes longer.

8.3.2 Why does everyone take the zigzag?

In the 4,000-car example, everyone's switching to the zigzag is no coincidence. The zigzag route is faster than the old routes however the remaining traffic is distributed. To see this, let z be the number of cars on the zigzag and split the remaining $4{,}000 - z$ cars evenly between the two old routes.

- Each old route carries $(4{,}000 - z)/2 = 2{,}000 - z/2$ cars.
- The flow on s→A is the sum of those on the upper route and those on the zigzag: $2{,}000 - z/2 + z = 2{,}000 + z/2$. The time is $20 + z/200$ minutes. The same holds for B→t.
- Old route: one narrow road and one wide road, $20 + z/200 + 45 = 65 + z/200$ minutes.

- Zigzag: two narrow roads and the shortcut, $2 \cdot (20 + z/200) + 0 = 40 + z/100$ minutes.

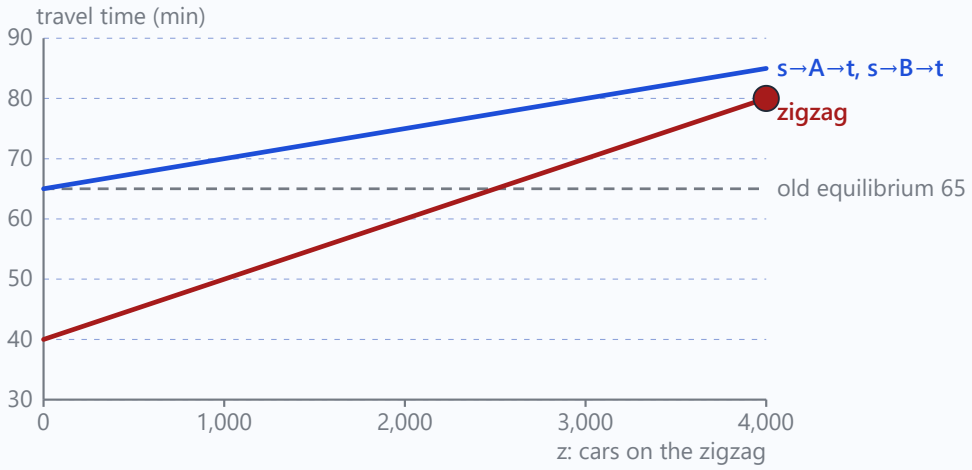


Figure 8.13: Travel times against the number of cars z on the zigzag. The blue line is the old routes ($65 + z/200$), the red line the zigzag ($40 + z/100$). The red line is below the blue for every z; at $z = 4{,}000$ everyone experiences 80 minutes. The dashed line is the old equilibrium (65).

Figure 8.13 shows the two lines. The zigzag would be slower than the old routes only if $40 + z/100 \geq 65 + z/200$; that means $z/200 \geq 25$, that is, $z \geq 5{,}000$. But there are only 4,000 cars. So the zigzag is faster in every case.

The same conclusion can be reached by a shorter route. The second half of the zigzag is B→t; even when it is most crowded it takes at most $4{,}000/100 = 40$ minutes. The first half of the old lower route, s→B, always takes 45 minutes. So going from s to B by way of A (s→A takes at most 40 minutes) is always faster than taking the highway directly (45 minutes). Symmetrically, going from A to t by way of B is always faster. The zigzag is a **dominant choice** for every driver: it is better whatever the others do.

The resemblance to the Prisoner's Dilemma is clear. In that game defection is the dominant strategy for each player, but when everybody defects everybody ends up worse off. Here too the zigzag is dominant for everyone, but each driver who switches loads the narrow roads. At $z = 4{,}000$ everyone experiences 80 minutes; had the shortcut never existed, everyone would have experienced 65.

8.3.3 The system optimum of the network with the shortcut

The shortcut itself is not bad. What is bad is its selfish use. To see this, compute the system optimum of the network with the shortcut: how should a planner distribute the traffic to make the average time as small as possible?

Example 8.10

1. By symmetry the two old routes are given equal numbers of cars (y); z cars are on the zigzag, $2y + z = 4{,}000$. The flow on the narrow roads is $a = y + z$.
2. The total latency is written edge by edge. On each narrow road a cars spend $a/100$ minutes: the two narrow roads together contribute $2 \cdot a \cdot a/100 = a^2/50$. On the wide roads a total of $2y$ cars spend 45 minutes: $90y$. The shortcut contributes 0. Since $y = 4{,}000 - a$, $C(a) = a^2/50 + 90\,(4{,}000 - a)$.
3. Set the derivative to zero: $C'(a) = a/25 - 90 = 0$, that is, $a = 2{,}250$. From this $y = 1{,}750$ and $z = 500$.
4. Total: $2{,}250^2/50 + 90 \cdot 1{,}750 = 101{,}250 + 157{,}500 = 258{,}750$ car-minutes. The average is $258{,}750 / 4{,}000 \approx 64.7$ minutes.
5. In this distribution the narrow roads take 22.5 minutes. The 500 drivers on the zigzag experience $22.5 + 22.5 = 45$ minutes, and the 3,500 drivers on the old routes $22.5 + 45 = 67.5$ minutes.

At the optimum the zigzag is given a small share (500 cars) and the average time falls slightly from 65 to 64.7. Used correctly, the shortcut brings a small improvement. But this distribution is not an equilibrium: drivers on the old routes experience 67.5 minutes and those on the zigzag 45, so all the drivers on the old routes want to switch to the zigzag. Table 8.7 compares the four cases.

	Selfish equilibrium	Optimum
No shortcut	20 + 45 = 65	65
With shortcut	40 + 0 + 40 = 80	≈ 64.7

Table 8.7: Average travel time in the Braess network (minutes).

The table answers the two questions at the beginning of the game. When the shortcut opens, trips do not get shorter but longer (65 → 80). When everyone picks his or her own fastest route, the average time is not the smallest possible (80 > 64.7). The answer of the theory to both questions is “no”.

8.3.4 The lesson of the paradox and real examples

Definition 8.6

Braess's paradox is the phenomenon that adding a new link to a network can, because of selfish route choice, increase the equilibrium latency, even the latency of all users (Braess, 1968).

The logic of the paradox is this: the new road looks attractive individually, but the traffic pattern the users form together moves to a worse equilibrium. The converse is also true: closing a road can sometimes speed everybody up. In the Braess network, closing A→B lowers everyone's time from 80 to 65 minutes.

The paradox has not stayed on paper. There are a few frequently cited examples:

- **Stuttgart (1969).** After a new road investment in the city center, traffic is said to have worsened, and to have improved when a street was closed (Knödel, 1969).
- **New York (1990).** When 42nd Street was closed to traffic on Earth Day, the expected congestion is reported not to have materialized (Kolata, 1990).
- **Seoul.** The elevated highway over the Cheonggyecheon stream was removed in the early 2000s and turned into an urban park. It is often cited that city traffic afterwards did not worsen as much as expected.
- **Mechanical and electrical networks.** A similar paradox has been demonstrated in networks built from springs and resistors: cutting a spring can raise a hanging weight (Cohen and Horowitz, 1991).
- **Laboratory experiments.** When subjects chose routes in a computer environment, they drifted over time to the slower equilibrium predicted by the paradox once a shortcut was added (Rapoport et al., 2009).

These examples must be interpreted carefully. Real networks are far more complicated; when a street is closed, some drivers switch to public transport and some change their travel time. The examples are therefore not conclusive proof of the paradox but an indication that it is possible. The lesson is clear, though: building or closing a road must be judged not only by capacity but by the equilibrium the drivers will reach. There are also ways to reduce the effect of the paradox: pricing, access restrictions or centralized routing. These are discussed in 8.6.

8.3.5 Code trace: best_subnet()

Braess's paradox makes the network designer ask a new question: which edges of the given network should be closed to lower the equilibrium time the most? The function below solves a simple version of this question: it tries closing a single edge.

```
def best_subnet(net, r):
  best, cut = eq_time(net, r), None
  for e in net:
    sub = [k for k in net if k != e]
    d = eq_time(sub, r)   # inf if no path
    if d < best:
      best, cut = d, e
  return cut, best
```

The function uses a helper function called `eq_time(net, r)`. It returns the common time of the paths used in the Wardrop equilibrium of the given subnetwork. In the script this time is computed by minimizing the Beckmann potential (8.2.5). If no path from s to t remains, it returns infinity (`inf`). The code works line by line as follows:

- `best` is the smallest equilibrium time found so far; at first it is the time of the network with nothing removed. `cut` holds which edge was removed; at first it is `None` (none).
- The loop tries each edge `e` of the network in turn.
- `sub` is the subnetwork consisting of all edges other than `e`.
- `d` is the equilibrium time of this subnetwork.
- If `d` is *strictly* smaller than the best so far, the candidate is updated.
- At the end the edge to remove and the new equilibrium time are returned.

Edge removed	Remaining routes and equilibrium	`d` (min)	`d < best`?	`best, cut`
— (start)	Everyone on the zigzag	80	—	80, None
s→A	Only s→B→t: *40 + 45*	85	no	80, None
A→t	Zigzag and lower; the zigzag dominates	80	no (equal)	80, None
s→B	Upper and zigzag; the zigzag dominates	80	no (equal)	80, None
B→t	Only s→A→t: *40 + 45*	85	no	80, None
A→B	The network before Braess: 2,000 / 2,000, *20 + 45*	65	yes	65, A→B

Table 8.8: Trace of `best_subnet(braess, 4000)`.

Two rows of the table need a separate explanation. When A→t is closed, two routes remain: the zigzag (s→A→B→t) and the lower route (s→B→t). Both use B→t, so there are 4,000 cars on B→t and it takes 40 minutes. The first part of the zigzag, s→A, takes at most 40 minutes, while the first part of the lower route, s→B, takes 45; the zigzag is again dominant. Everyone switches to the zigzag and the time is *40 + 0 + 40 = 80* minutes. When s→B is closed, the symmetric situation arises. In these two rows `d` equals the old best; since the code looks for strict improvement, the candidate does not change.

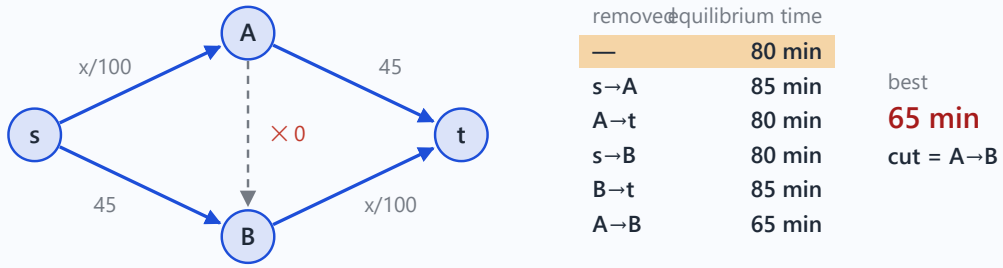


Figure 8.14: The result of `best_subnet`. On the left the network (A→B dashed and crossed out: the edge to be closed), in the middle the equilibrium time for each removal attempt, on the right the result: `best = 65`, `cut = A→B`.

Figure 8.14 shows the result: A→B should be closed, and the equilibrium time falls from 80 to 65 minutes. The code has a limitation: it tries removing only *one* edge. In general it can be better to remove several edges together. In a network with m edges, trying all subnetworks requires 2^m equilibrium computations. For a network with 30 edges this is more than a billion; real city networks have thousands of edges. The designer's question “which edges should I keep open?” is therefore hard in a general network.

8.3.6 Network design for selfish users

The question can be stated formally: for which subnetwork of the given network is the time at the Wardrop equilibrium smallest? Here, closing a road rather than building one is also a design decision. Two results outline the difficulty and the limits of this problem.

Hardness. In 2006 Tim Roughgarden showed that this problem is computationally hard (Roughgarden, 2006). With linear latencies, even closing no edge at all is at most *4/3* times worse than the best choice. But guaranteeing anything better than this efficiently is NP-hard: an efficient algorithm cannot be expected to find, on every network, a subnetwork noticeably better than the option of “closing nothing”. In the Braess example the ratio is below this bound: *80/65 ≈ 1.23 < 4/3 ≈ 1.33*.

Topology. In 2006 Igal Milchtaich determined in which networks the paradox can never occur (Milchtaich, 2006). In **series-parallel networks** Braess's paradox never occurs, however the latencies are chosen. A series-parallel network is one built by joining edges one after another (in series) or side by side (in parallel). The two-route example is of this kind: two edges are connected in parallel between s and t. In other networks the paradox can occur with suitable latencies.

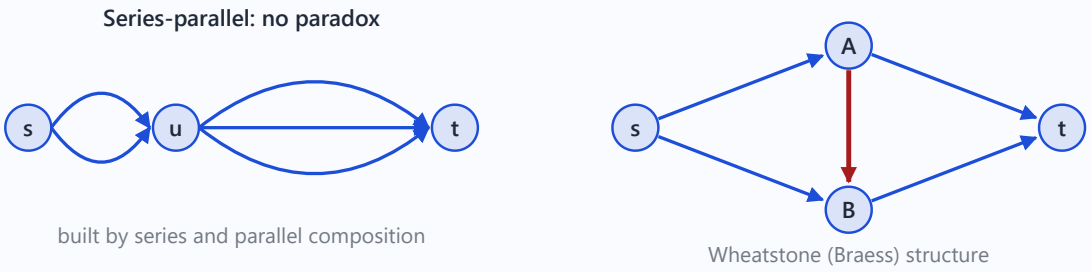


Figure 8.15: On the left a series-parallel network: two parallel edges between s and u, three parallel edges between u and t, the two groups joined in series. On the right the Wheatstone structure: the A→B bridge makes the network impossible to build by series and parallel composition.

Figure 8.15 compares the two structures. The Braess network is the **Wheatstone network** on the right; it takes its name from the Wheatstone bridge in electrical circuits. The A→B bridge “crosses” two parallel routes. Such a cross cannot be built by series and parallel composition. By Milchtaich's result, a necessary and sufficient condition for a two-terminal network to be immune to the paradox

is that it contain no Wheatstone structure, which is the same as the network being series-parallel. Networks with cycles, or complete graphs, generally contain a Wheatstone structure and give no such guarantee.

8.3.7 Common mistakes about Braess's paradox

Common mistakes

- **Taking the shortcut to be a "bad road":** The shortcut itself is not bad; it is used at the optimum and lowers the average slightly (64.7). The problem is selfish use.
- **Forgetting flow conservation:** The flow on s→A is not only those on the upper route; those on the zigzag use that road too ($y + z$).
- **Assuming everyone always switches to the zigzag:** With 6,000 cars the zigzag is not dominant; the equilibrium splits the traffic (3,000 on the zigzag, 1,500 on each old route).
- **Assuming the paradox appears in every network and at every traffic level:** It never appears in series-parallel networks. Even in the Braess network, when traffic is very light the shortcut can be genuinely useful; the paradox arises in a certain range of traffic.
- **Limiting the closing decision to a single edge:** The best subnetwork may require closing several edges; trying a single edge is only a heuristic.

8.4 Congestion games

The nonatomic traffic model suits thousands of small users. But in some cases the number of players is small and each one's effect is noticeable: a few large freight companies, a few data centers, or a few applications sending packets over a network. The model in which the players can be counted one by one is called the **atomic** model. For this model too there is a powerful tool showing that selfish choices reach an equilibrium: Rosenthal's potential.

8.4.1 Congestion games: atomic and nonatomic

Definition 8.7

A **congestion game** consists of the following components:

- n players and a set of resources E (for example the edges of a network).
- The strategy of each player is a subset of the resources (for example an s–t path, that is, the edges of that path).
- The cost $c_e(n_e)$ of each resource e depends only on the number n_e of players using it and increases with this load.
- A player's cost is the sum of the costs of the resources chosen: $c_i(s) = \sum_{e \in s_i} c_e(n_e)$.

The important point in the definition is that the cost of a resource depends on *how many* use it, not on *who* uses it. It makes no difference whether a path carries Ali and Ayşe or Can and Deniz; if there are two users, the cost is $c_e(2)$. While the players try to reduce their own delays, the total network cost may rise; Braess's paradox appears in the atomic model too (8.4.3).

The atomic model and the nonatomic model are two faces of the same idea. Table 8.9 summarizes the differences.

	Atomic	**Nonatomic**
Players	finite, n of them	continuous flow
Effect	one player raises the load by 1	one player is neglected
Equilibrium	pure Nash	Wardrop
Potential	Rosenthal (sum)	Beckmann (integral)

Table 8.9: Atomic and nonatomic congestion models.

In the atomic model, when a player switches paths, the user count of every edge on the new path rises by one. The player must therefore count himself or herself when computing the cost of the new path: a player moving to an edge currently used by 2 people pays $c_e(3)$, not $c_e(2)$. This small detail is the source of the most frequent mistake in atomic calculations.

In 1973 Robert Rosenthal showed an important property of congestion games: every finite congestion game has a potential function, so a **pure Nash equilibrium** always exists (Rosenthal, 1973). This is a remarkable result. In general games a pure equilibrium may not exist; in Rock–Paper–Scissors only a mixed equilibrium exists. In congestion games mixed strategies are not needed.

8.4.2 Rosenthal's potential function

Definition 8.8

In a strategy profile s, let n_e be the number of players using resource e. **Rosenthal's potential** is

$$\Phi(s) = \Sigma_e \left[c_e(1) + c_e(2) + \ldots + c_e(n_e) \right]$$

On each edge the costs from 1 up to the number of users n_e are added. This is not the total cost. In the total cost each of the n_e users on that edge pays $c_e(n_e)$, and the edge contributes $n_e \cdot c_e(n_e)$. In the potential, the users are imagined to enter the edge one after another: the first to arrive "pays" $c_e(1)$, the second $c_e(2)$, the last $c_e(n_e)$. This is the atomic counterpart of the "area under the curve" in the Beckmann potential: a sum replaces the integral.

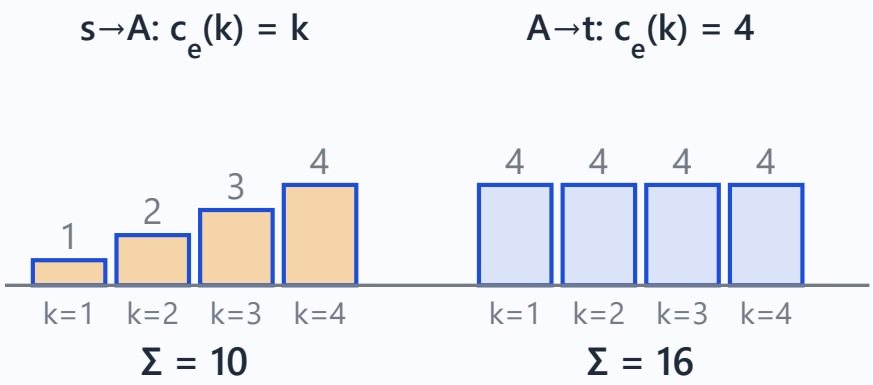


Figure 8.16: The contribution of Rosenthal's potential from two edges. On edge s→A, $c_e(k) = k$: with four users the contribution is $1 + 2 + 3 + 4 = 10$. On edge A→t, $c_e(k) = 4$: the contribution is $4 + 4 + 4 + 4 = 16$.

Figure 8.16 shows this sum as bars. On edge s→A, whose cost grows with the number of users, the bars rise like a staircase; on edge A→t, whose cost is constant, they are all the same height.

Example 8.11

The atomic version of the Braess network has 4 drivers. Edge costs: $c_e(k) = k$ on s→A and B→t (k: number of users), 4 on s→B and A→t, 0 on A→B. All 4 drivers are on the path s→A→t (upper).

1. There are 4 users on s→A: the contribution is $1 + 2 + 3 + 4 = 10$.
2. There are 4 users on A→t: the contribution is $4 + 4 + 4 + 4 = 16$.
3. s→B, B→t and A→B are empty: the contribution is 0.
4. $\Phi = 10 + 16 = 26$.

5. For comparison, the total cost: each driver pays $c(s \to A) + c(A \to t) = 4 + 4 = 8$; the total is $4 \times 8 = 32$. The potential (26) differs from the total cost (32).

What makes the potential valuable is the following identity.

Theorem 8.2 (Rosenthal)

In a congestion game, if a single player i changes strategy (the others fixed), the change in the potential is exactly equal to the change in that player's cost:

$$\Delta\Phi = c_i(\text{new}) - c_i(\text{old})$$

Therefore every finite congestion game has a pure Nash equilibrium, and any sequence of improving moves always stops after a finite number of steps.

Proof. Let player i move from the old strategy P to the new strategy P'. The edges fall into three groups, and we look at how Φ changes in each group.

(1) Edges on both P and P': i keeps using these edges and their user counts do not change. The contribution of these edges to Φ stays the same. The cost i pays on these edges also stays the same.

(2) Edges only on P: i leaves these edges and the user count falls from n_e to $n_e - 1$. The last term $c_e(n_e)$ of the sum for this edge is deleted. This term is the cost i paid on that edge in the old state.

(3) Edges only on P': i enters these edges and the user count rises from n_e to $n_e + 1$. A new term $c_e(n_e + 1)$ is added to the sum. This term is the cost i will pay on that edge in the new state.

Combining the three groups, the change in Φ is “the added terms minus the deleted terms”; this is i's new cost minus the old cost. The common edges cancel in the difference because they are the same on both sides. Thus $\Delta\Phi = c_i(\text{new}) - c_i(\text{old})$.

If a player makes a move that lowers his or her cost ($c_i(\text{new}) < c_i(\text{old})$), Φ strictly decreases. Since the game is finite, the number of strategy profiles is finite, and Φ therefore takes only finitely many values. A strictly decreasing sequence cannot go on forever among finitely many values. So the improving moves stop at some point. Where they stop, no player can lower his or her cost alone; by definition this is a pure Nash equilibrium. ■

The proof has one more consequence: the profile that gives the smallest value of Φ over all profiles is also a pure Nash equilibrium. For there, a move lowering any player's cost would lower Φ still further, which contradicts being at the minimum.

8.4.3 Best-response dynamics and Φ step by step

To see the theorem at work, **best-response dynamics** is applied to the atomic Braess network of Example 8.11. Drivers move in the order 1, 2, 3, 4. The driver whose turn it is picks the route with the smallest cost, the others being fixed. There are three routes: upper (s→A→t), lower (s→B→t) and zigzag (s→A→B→t). In case of a tie the driver picks the first in this order, but moves only if there is a strict improvement.

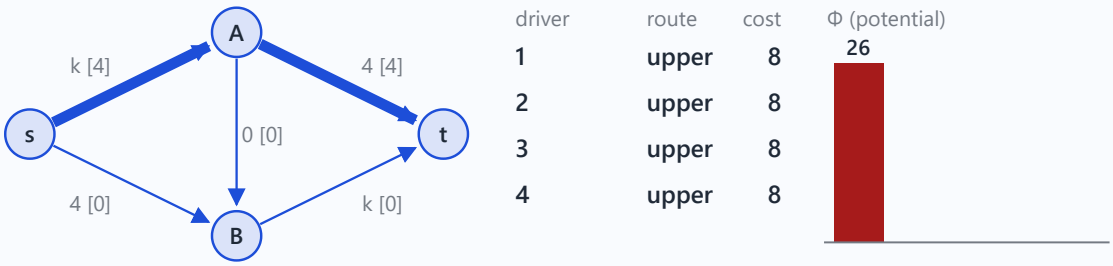


Figure 8.17: Start: all four drivers on the upper route. The edge labels show the cost function and, in square brackets, the number of users. Everyone's cost is $4 + 4 = 8$, $\Phi = 26$.

Figure 8.17 shows the initial state. The dynamics proceed in four moves as follows.

Example 8.12

1. **Driver 1** (upper, 8). Staying on the upper route costs 8. Moving to the lower route, s→B costs 4 and B→t has a single user (himself or herself): $4 + 1 = 5$. Moving to the zigzag, the user count on s→A is still 4 (the driver is already there): $4 + 0 + 1 = 5$. Lower and zigzag tie; the first in the order, lower, is picked. The cost falls from 8 to 5. Φ: 26 → 23 ($\Delta = -3$).
2. **Driver 2** (upper, now 7: 3 people on s→A). Upper 7. Lower: s→B 4, and on B→t 2 people with driver 1: $4 + 2 = 6$. Zigzag: s→A 3, A→B 0, B→t 2: $3 + 0 + 2 = 5$. The zigzag is best. The cost falls from 7 to 5. Φ: 23 → 21 ($\Delta = -2$).
3. **Driver 3** (upper, 7). Upper 7. Lower: $4 + 3 = 7$. Zigzag: $3 + 0 + 3 = 6$. The zigzag is best. The cost falls from 7 to 6. Φ: 21 → 20 ($\Delta = -1$).
4. **Driver 4** (upper, 7). Lower: $4 + 4 = 8$. Zigzag: $3 + 0 + 4 = 7$. There is nothing better; the driver stays.
5. **Second round.** Driver 1 (lower, 7): upper $4 + 4 = 8$, zigzag $4 + 0 + 3 = 7$ (on entering s→A there would be 4 people there); no strict improvement. Drivers 2 and 3 (zigzag, 6): upper 7, lower 7; they stay. Driver 4 (upper, 7): stays again. Nobody can improve: a pure Nash equilibrium, $\Phi = 20$.

At each step the drop in Φ equals the gain of the driver who moved: 3, 2, 1. This confirms the identity of the theorem with numbers. In driver 1's move, for example, one term is deleted from each of the edges s→A and A→t (4 and 4, total 8: the old cost), and one term is added to each of the edges s→B

and B→t (4 and 1, total 5: the new cost). $\Delta\Phi = 5 - 8 = -3$.

Move	Current	upper / lower / zigzag	Choice	Φ
Start	everyone upper, 8	—	—	26
Driver 1	upper, 8	8 / 5 / 5	lower	26 → 23
Driver 2	upper, 7	7 / 6 / 5	zigzag	23 → 21
Driver 3	upper, 7	7 / 7 / 6	zigzag	21 → 20
Driver 4	upper, 7	7 / 8 / 7	stays	20
Round 2	(lower, zigzag, zigzag, upper)	nobody can improve	—	20

Table 8.10: Trace of the best-response dynamics (option costs: upper / lower / zigzag).

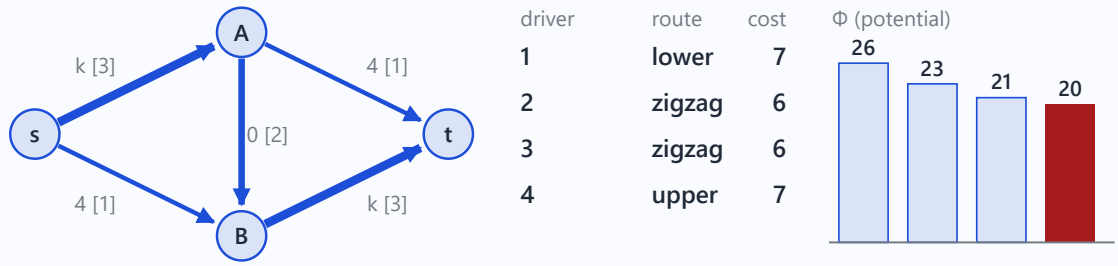


Figure 8.18: The end of the dynamics: the equilibrium (lower, zigzag, zigzag, upper). The costs are 7, 6, 6, 7; the bars on the right show the fall of Φ from 26 to 23 to 21 to 20.

Figure 8.18 shows the equilibrium. The total cost is $7 + 6 + 6 + 7 = 26$. Had edge A→B never existed, the drivers would have split 2 + 2: 2 people on the upper route (s→A 2, A→t 4: cost 6) and 2 on the lower route (s→B 4, B→t 2: cost 6). The total would have been 24. The shortcut has raised the total cost from 24 to 26. This is the atomic version of Braess's paradox.

One more detail matters: the equilibrium that the dynamics reached is not the only equilibrium of the game. Scanning all profiles, and ignoring the drivers' names, four pure equilibria are found: (lower, zigzag, zigzag, upper), (lower, zigzag, zigzag, zigzag), (zigzag, zigzag, zigzag, upper) and the profile in which all four are on the zigzag. Their total costs are 26, 29, 29 and 32 respectively. In all four $\Phi = 20$, which is the smallest value of the potential. Which equilibrium is reached depends on the order of moves and on the tie-breaking rule.

Example 8.13

At the equilibrium (lower, zigzag, zigzag, upper), compute Φ directly and look at the effect of driver 4 moving to the zigzag.

1. User counts: s→A is used by two zigzag drivers and one upper driver (3 people); B→t by the lower driver and two zigzag drivers (3 people); A→t only by the upper driver (1); s→B only by the lower driver (1); A→B by the two zigzag drivers (2).
2. $\Phi = (1 + 2 + 3) + (1 + 2 + 3) + 4 + 4 + (0 + 0) = 20$.

3. If driver 4 moves to the zigzag: on s→A the driver is already there (3 people, 3); A→B 0; on B→t the user count rises to 4 (4). The new cost is $3 + 0 + 4 = 7$. The old cost was also 7.
4. Since the cost does not change, $\Delta\Phi = 0$. Check: 4 is deleted from A→t, 0 is added to A→B and 4 to B→t; the difference is 0.

8.4.4 Potential games: what have we gained?

Rosenthal's idea generalizes beyond congestion games. In 1996 Dov Monderer and Lloyd Shapley gave this generalization a name (Monderer and Shapley, 1996).

Definition 8.9

A game is called an **(exact) potential game** if there is a function Φ that reproduces exactly the cost change of every player under every unilateral deviation: for every i, every s_i, s_i' and every s_{-i}

$$\Phi(s_i', s_{-i}) - \Phi(s_i, s_{-i}) = c_i(s_i', s_{-i}) - c_i(s_i, s_{-i})$$

Φ is called the **potential function**.

The potential function gathers the selfish moves of all the players into a single “common measure”. Every player wants to lower his or her own cost; in a potential game every move that does so also lowers the common measure. The players are, without knowing it, working to minimize the same function. This viewpoint brings several gains:

- **Existence.** The state that gives the smallest value of Φ is a pure Nash equilibrium; an equilibrium always exists.
- **Convergence.** A sequence of improving moves cannot go on forever. This property is called the **finite improvement property**. Every best-response dynamics stops.
- **Learning.** The players do not need to “know” the equilibrium. Best-responding in turn is enough to reach it. This is a strong explanation of why equilibria can arise in real life.
- **Efficiency.** The equilibrium that minimizes Φ is one way to choose a good equilibrium among the equilibria, because Φ is closely related to the total cost. But this is no guarantee: in the network of Example 8.13, $\Phi = 20$ in all four equilibria, while their total costs range from 26 to 32.
- **Generality.** Every (exact) potential game is equivalent to a congestion game (Monderer and Shapley, 1996). In other words, congestion games are not merely a “typical” example of potential games; they are exactly equivalent to them.

There is also a limit: **speed**. The dynamics are guaranteed to stop, not to stop quickly. Φ decreases by at least some amount at each step, but its initial value can be very large and the number of steps can grow exponentially with the number of players. In 2004 Alex Fabrikant, Christos Papadimitriou and

Kunal Talwar showed that finding a pure equilibrium in general congestion games is **PLS-complete** (Fabrikant, Papadimitriou, Talwar, 2004). PLS is the complexity class of problems of finding a "locally optimal" solution. Being PLS-complete means that no fast method is known for finding an equilibrium in some congestion games. In simple cases (for example symmetric network games, in which all players go from the same source to the same destination) the equilibrium can be found quickly by reduction to a flow problem.

8.4.5 Common mistakes with potentials

Common mistakes

- **Taking the potential to be the total cost:** On an edge with $c_e(k) = k$ and 4 users, the contribution to Φ is $1 + 2 + 3 + 4 = 10$, while the contribution to the total cost is $4 \cdot 4 = 16$.
- **Not counting oneself on the new route:** A player moving to an edge raises that edge's user count by one. A player moving to an edge currently used by 3 people pays $c_e(4)$.
- **Changing a common edge twice:** A player who is on an edge in both the old and the new route stays there; the user count of that edge does not change (had driver 1 moved to the zigzag instead of the lower route, 4 people would have stayed on s→A).
- **Taking the equilibrium to be a maximum of Φ:** Cost-reducing moves decrease Φ; an equilibrium is a (local) minimum of Φ.
- **Taking stopping to mean speed:** The finite improvement property says that the dynamics stop, not in how many steps; in general this number can be exponential.

8.5 Network formation

Up to here the network was taken as given; the users decided only which route to take. In network formation games the players build the network itself. On the Internet, network operators decide for themselves which links to set up with each other; in social networks people choose whom to befriend; in peer-to-peer (P2P) file-sharing networks every computer chooses which computers to connect to. The basic concept behind these decisions is externality.

8.5.1 Network externalities: negative and positive

Definition 8.10

Externality: a user's decision changes the cost or the benefit of others, but the user does not include this effect in his or her own calculation. If the effect harms others it is a *negative* externality; if it benefits them, a *positive* one.

Negative externality: congestion. Every additional driver entering a road slows each of the other x people on the same road by $\ell'(x)$, imposing a total delay of $x \cdot \ell'(x)$. In the two-route example, with 2,000 cars on Route 1 ($\ell_1' = 1/100$) this burden is $2{,}000 \times 1/100 = 20$ car-minutes. The new driver sees his or her own 25 minutes, not the 20 car-minutes imposed on the others. The result: the crowded route is used more than it should be. This is the reason the equilibrium deviates from the optimum in 8.2.4. The remedy is a **congestion toll**: if the driver is charged an amount equal to the delay imposed on others, the driver takes this effect into account too (8.6.2).

Positive externality: network effect. For some networks the value of joining grows with the number of participants. A telephone owned by one person is useless; two people can call each other; in a network of a hundred people every user can reach 99 others. Messaging apps and social networks are like this too. In a network with n users there are $n(n-1)/2$ possible connections: 45 with 10 users, 4,950 with 100 users. The claim that the value of a network therefore grows as n^2 is called **Metcalfe's law**. This claim is not an exact law but a rough estimate: not all connections are equally valuable, and most users communicate with only a small fraction of those in the network.

The network effect also involves an externality: when a new user joins the network, he or she adds value for the others (they can now reach one more person), but when deciding whether to join looks only at his or her own benefit. Adoption therefore stays slow until a **critical mass** is passed: joining a network used by few people appeals to nobody.

In a network formation game both externalities are present. A link shortens the distances not only of the one who builds it: everyone who passes through that link benefits (a positive externality). But only the one who buys the link pays for it. The players may therefore build fewer links than is best for society.

8.5.2 Network formation games

In network formation models a link is simply called a **link** or an **edge**. Each link has a construction cost; being well connected, that is, being at a short distance from the others, is valuable. This is the basic tension of the model: balancing the cost of links against reach. There are two basic models.

Definition 8.11

The one-sided network formation game (Fabrikant et al., 2003). There are n players. Each player i buys links to the nodes he or she wants; each link costs α. Whoever buys it, a link is used by everyone in both directions. The player's cost:

$$cost_i = \alpha \cdot (\text{number of links bought by } i) + \Sigma_j \, distance(i, j)$$

The distance is the number of links on the shortest path between two nodes; if the network is disconnected, the distance to an unreachable node is infinite. The solution concept is Nash equilibrium. The **social cost** is the sum of the costs of all players.

The two-sided model (Jackson and Wolinsky, 1996). A link is formed only if the players at both ends want it; if either wants, it is cut. The solution concept is **pairwise stability**.

The two models correspond to different real-life situations. The one-sided model resembles a website linking to other sites, or a network operator leasing a line on its own: the one who builds the link pays, and the other side's consent is not needed. The two-sided model resembles friendship, or a mutual **peering** agreement between two network operators: both sides must agree.

Three questions are asked in both models. Which networks are in equilibrium? Which network is best for society? How different can the two be? These questions are the network-formation counterpart of the question “equilibrium or optimum?” in selfish routing.

8.5.3 Two-sided links: pairwise stability

Definition 8.12

A network that satisfies the following two conditions is called **pairwise stable**:

1. no player gains by cutting one of his or her own links;
2. no missing link would make both of its endpoints better off (if one gains strictly and the other does not lose, the link is formed).

The definition differs slightly from Nash equilibrium. In Nash equilibrium the deviation of a single player is examined. In pairwise stability, cutting a link is one person's decision, but forming a link is the joint decision of two people. In evaluating a link that has not been formed, the gains of both ends are considered.

For a concrete example take $n = 4$ players (A, B, C, D). A player's benefit depends on the distance to the others as follows: 4 from each player at distance 1, 2.2 from one at distance 2, and 1 from one at distance 3. The cost of each link is 2 for each of the players at its two ends. The player's net benefit is

the sum of the distance benefits minus the link costs. Figure 8.19 shows the four networks to be compared.

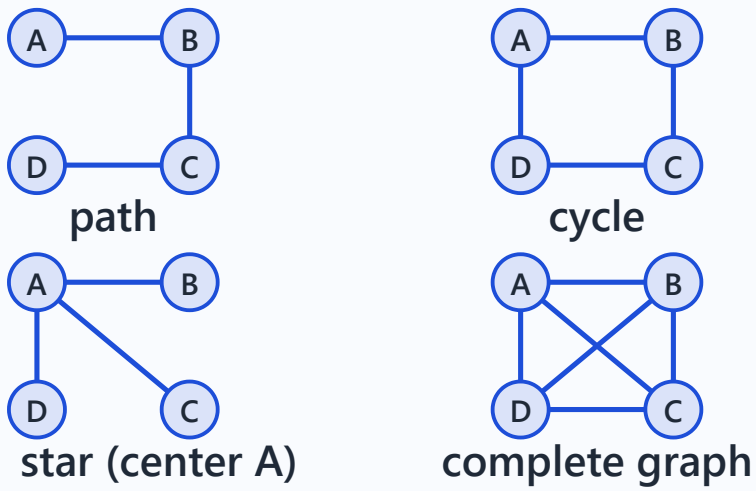


Figure 8.19: Four networks on four players: the path (A–B–C–D), the cycle, the star (center A) and the complete graph.

Example 8.14

1. **Path A–B–C–D.** A's distances: B 1, C 2, D 3. Benefit $4 + 2.2 + 1 = 7.2$; one link, cost 2; net 5.2. B's distances: A 1, C 1, D 2. Benefit $4 + 4 + 2.2 = 10.2$; two links, cost 4; net 6.2. By symmetry C gets 6.2 and D 5.2. Total 22.8.
2. **Is the path pairwise stable?** Add the link A–D. For A the distance to D falls from 3 to 1: the benefit rises by $4 - 1 = 3$ and the cost is 2; net +1. The same holds for D. Both ends gain: the link is formed. The path is not pairwise stable.
3. **Cycle.** Everyone has two neighbors (distance 1) and one opposite node (distance 2). Benefit $4 + 4 + 2.2 = 10.2$, cost of two links 4; net 6.2. Total 24.8. If a player cuts a link the cycle becomes a path and that player becomes an end of the path: net falls to 5.2; the player loses. If a diagonal is added (A–C), at the two ends the distance falls from 2 to 1: the benefit rises by $4 - 2.2 = 1.8$ and the cost is 2; net −0.2. Nobody wants to add it. The cycle is pairwise stable.
4. **Star (center A).** The center has three neighbors: benefit 12, cost 6; net 6. Each leaf has one neighbor (the center) and two leaves at distance 2: benefit $4 + 2.2 + 2.2 = 8.4$, cost 2; net 6.4. Total $6 + 3 \times 6.4 = 25.2$. If the center cuts a link that leaf is disconnected from the network, and the center loses 4 in benefit while saving only 2. If a leaf cuts its link it loses all its benefit. If two leaves formed a link each would gain 1.8 but pay 2: they do not add it. The star is pairwise stable.

5. **Complete graph.** Everyone is linked to everyone: benefit 12, cost of three links 6; net 6. Total 24. If a player cuts a link, the distance to that neighbor rises from 1 to 2 (through the others): the player loses 1.8 in benefit but saves 2 in cost; net +0.2. The complete graph is not pairwise stable.

Network	Net benefits	Total	Pairwise stable?
Path	5.2 / 6.2 / 6.2 / 5.2	22.8	No: A–D is added (each gains +3 in benefit, −2 in cost)
Cycle	6.2 (all)	24.8	Yes
Star	6 / 6.4 / 6.4 / 6.4	25.2	Yes; the network with the highest total
Complete graph	6 (all)	24	No: the player who cuts a link gains 2 and loses 1.8

Table 8.11: Net benefits and pairwise stability in four-player networks.

Table 8.11 gives two lessons together. The first is that there can be more than one pairwise stable network: both the cycle and the star are stable. Scanning all networks on four players (64 possible sets of links), only the four stars (with different centers) and the three cycles (with different node orders) turn out to be pairwise stable. The second is that a stable network need not coincide with the network with the highest total. The star has the highest total among all networks (25.2). The cycle is stable but 0.4 behind the star. If the players reach the cycle they do not leave it on their own; the move of a single player or a single pair cannot turn the cycle into the star.

8.5.4 One-sided links: a network formation game step by step

In the one-sided model the example is as follows: $n = 4$ players, link price $\alpha = 1.5$. Each player chooses which links to buy; whoever buys a link, everyone uses it. A player's cost is 1.5 times the number of links bought plus the sum of the distances to the others. If the network is disconnected the cost is infinite; so no player makes a move that disconnects the network.



Path 1–2–3–4 (i buys i+1)

costs 7.5 · 5.5 · 5.5 · 6

social cost 24.5

Figure 8.20: The path network 1–2–3–4; each player i buys the link $(i, i + 1)$, and 4 buys none. Costs above the nodes: 7.5 · 5.5 · 5.5 · 6; social cost 24.5.

Figure 8.20 shows the path network, the starting point. Starting from this network, several candidate networks are checked in turn.

Example 8.15

1. **Path network.** Player 1 buys one link (1.5); the distances are 1 to 2, 2 to 3, 3 to 4, total 6. The cost is $1.5 + 6 = 7.5$. Player 2: one link (1.5), distances $1 + 1 + 2 = 4$, cost 5.5. Player 3: likewise 5.5. Player 4: buys no link, distances $3 + 2 + 1 = 6$, cost 6. Social cost $7.5 + 5.5 + 5.5 + 6 = 24.5$. The players at the ends are far from the others.
2. **A profitable deviation.** If player 1 buys a second link to 4, the network becomes a cycle. The distances become 1 to 2, 1 to 4, 2 to 3; total 4. The cost is $2 \cdot 1.5 + 4 = 7 < 7.5$. The path network is not a Nash equilibrium.
3. **Complete graph.** There are 6 links and every player is at distance 1 from the others. Social cost $6 \cdot 1.5 + 4 \cdot 3 = 9 + 12 = 21$. But if a player who has bought a link drops one, he or she saves 1.5; the distance to that neighbor rises only from 1 to 2 (through another node). Net gain $1.5 - 1 = 0.5$; the deviation is profitable. The complete graph is not an equilibrium.
4. **Star.** The center 1 buys the three links: its cost is $3 \cdot 1.5 + 3 = 7.5$. The cost of each leaf is $0 + 1 + 2 + 2 = 5$. If a leaf buys a link to another leaf, its distance to that leaf falls from 2 to 1: it gains 1, pays 1.5; its cost becomes 5.5 ($1 < 1.5$, unprofitable). If it buys links to both leaves it pays 3 and gains 2; again unprofitable. If the center drops a link the network is disconnected and its cost becomes infinite. Nobody deviates: the star is a Nash equilibrium. Social cost $7.5 + 3 \cdot 5 = 22.5$.
5. **Cycle.** Each player buys one link (for example 1 to 2; 2 to 3; 3 to 4; 4 to 1). Each pays $1.5 + 1 + 1 + 2 = 5.5$; social cost 22. A player who drops his or her link ends up at the end of a path: distances $1 + 2 + 3 = 6$, cost 6. One who adds a diagonal pays $3 + 3 = 6$. Both are worse than 5.5. The cycle is also a Nash equilibrium.

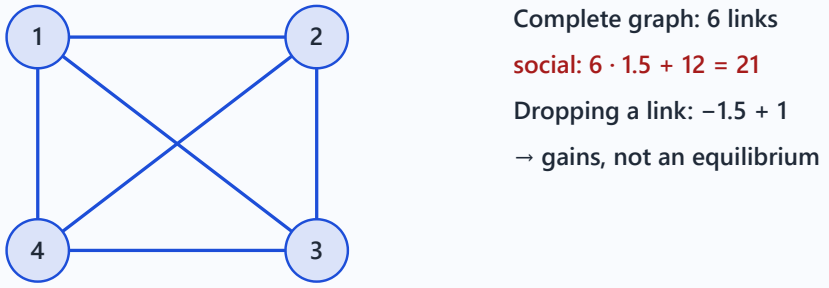
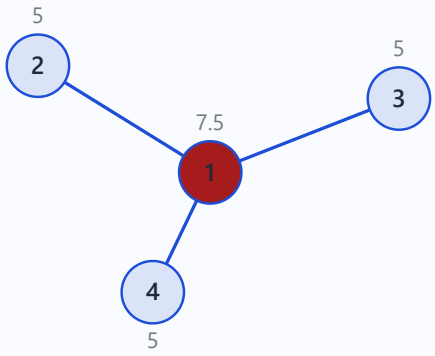


Figure 8.21: Complete graph: 6 links, social cost $6 \cdot 1.5 + 12 = 21$. A player who drops a link saves 1.5 and loses only 1 in distance; it is not an equilibrium.



Center 1 buys three links

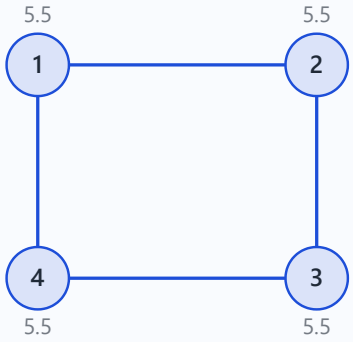
center 4.5 + 3 = 7.5; leaf 5

Leaf buys a link: 1.5 + 4 = 5.5

If the center drops one, the network is disconnect

→ Nash equilibrium (22.5)

Figure 8.22: Star: center 1 buys three links (cost 7.5), the leaves pay 5 each. A leaf that buys a link rises to 5.5; if the center drops a link the network is disconnected. Nash equilibrium, social cost 22.5.



4-cycle (each buys one link): 5.5

social cost 22

Script: all 4,096 profiles checked

equilibria: star 22.5 · cycle 22

optimum: complete graph 21

Figure 8.23: The 4-cycle: each player buys one link and pays 5.5; social cost 22. Checking all 4,096 strategy profiles shows that at $\alpha = 1.5$ the equilibria are the star (22.5) and the cycle (22); the optimum, the complete graph (21), is not an equilibrium.

Figure 8.21, Figure 8.22 and Figure 8.23 compare the three networks. The correctness of the result was verified by a brute-force scan. Each player's strategy is to which of the other three players to buy links: $2^3 = 8$ options. For four players there are $8^4 = 4{,}096$ strategy profiles. In every profile all deviations of every player were tried, and it was found that at $\alpha = 1.5$ the only networks in equilibrium are the star (social cost 22.5) and the cycle (22). The complete graph (21), which has the smallest social cost, is not an equilibrium.

The meaning of this result is the same as in selfish routing: equilibrium and optimum come apart. In the complete graph each link, by shortening the distance between two players, benefits *two* players (each is one step closer to the other, a social gain of 2). But the owner of the link looks only at his or her own gain (1) and compares it with the price of the link (1.5). The other side's gain is an externality.

8.5.5 Code trace: cost()

To compute a player's cost the distances to the others must be known. Since all links have equal length (1), the shortest distances are found by **breadth-first search** (BFS). BFS starts at the start node; it first visits all its neighbors (distance 1), then their not-yet-seen neighbors (distance 2), and so on.

```
def cost(i, links, alpha):
  dist = {i: 0}; queue = [i]
  while queue:
    u = queue.pop(0)
    for v in nbrs(links, u):
      if v not in dist:
        dist[v] = dist[u] + 1
        queue.append(v)
  bought = sum(1 for (a, b) in links if a == i)
  return alpha * bought + sum(dist.values())
```

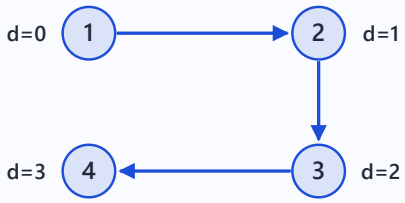
The function uses a helper function called `nbrs(links, u)`. It returns the neighbors of `u` without looking at who bought the link: whoever buys a link, everyone uses it. The list `links` consists of pairs `(a, b)`; `(a, b)` means that the link was bought by `a`. The code works line by line as follows:

- The dictionary `dist` holds the known distances; at first it contains only `i` (distance 0). `queue` is the list of nodes to be visited.
- Until the queue is empty, the node `u` at the front is removed.
- For each neighbor `v` of `u`: if `v` has not been seen before, its distance is set to one more than that of `u` and `v` is appended to the end of the queue.
- `bought` is the number of links bought by `i`: the pairs whose first element is `i` are counted.
- The cost is `alpha` times the number of links bought plus the sum of all the distances.

The code assumes the network is connected; if it is not, unreachable nodes never enter `dist` and the sum comes out wrongly small. In a real implementation one must return infinity when `len(dist) < n`.

Step	`u`	New neighbors	`queue`	`dist`
start	—	—	[1]	1:0
1	1	2 → 1	[2]	1:0, 2:1
2	2	3 → 2	[3]	1:0, 2:1, 3:2
3	3	4 → 3	[4]	1:0, 2:1, 3:2, 4:3
4	4	none (all visited)	[]	same
result	`bought = 1`, sum of distances 6, cost $1.5 \cdot 1 + 6 = 7.5$			

Table 8.12: Trace of `cost(1, links, 1.5)`; path network 1–2–3–4, player 1 has bought only the link (1, 2).



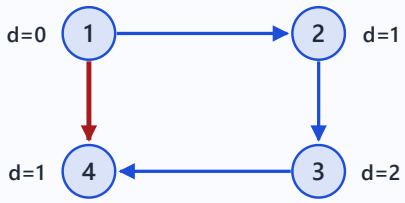
player i = 1
queue: []
dist: 1:0 2:1 3:2 4:3
Σ distances = 6
arrow: buyer → other end · red: new link

Figure 8.24: The end of the first BFS: the distances of player 1 in the path network are *d = 0, 1, 2, 3*; total 6. The arrows point from the buyer to the other end.

Figure 8.24 shows the end of the first call. Then a deviation is tried: player 1 also buys the link (1, 4). The same function is called again.

Step	`u`	New neighbors	`queue`	`dist`
start	—	—	[1]	1:0
1	1	2 → 1, 4 → 1	[2, 4]	1:0, 2:1, 4:1
2	2	3 → 2	[4, 3]	1:0, 2:1, 3:2, 4:1
3	4	none	[3]	same
4	3	none	[]	same
result	`bought = 2`, sum of distances 4, cost *1.5 · 2 + 4 = 7*			

Table 8.13: `cost(1, links, 1.5)` after the deviation; player 1 has bought the links (1, 2) and (1, 4).



player i = 1
queue: []
dist: 1:0 2:1 3:2 4:1
Σ distances = 4
arrow: buyer → other end · red: new link

Figure 8.25: The end of the BFS after the deviation: the red arrow is the new link (1, 4). Player 1's distances are 1 to 2 and 4, 2 to 3; total 4.

Figure 8.25 shows the distances shortened by the new link. Comparing the two calls checks the equilibrium: since *7 < 7.5*, the deviation is profitable and the path network is not a Nash equilibrium. In the second trace the order of the queue matters: because 4 enters the queue before 3, it comes out first. In BFS a node's distance is final the moment it is first seen; no shorter path can be found later, because nodes are visited in order of distance.

To try all deviations of a player, this function is called once for each possible set of links of that player. In a game with n players each player has 2^{n-1} sets of links, a number that grows quickly. Fabrikant and coauthors also showed that finding a player's best response is NP-hard in general (Fabrikant et al., 2003). Brute force suffices for small examples; large networks need approximate methods.

8.5.6 Equilibrium and optimum as the link price changes

When the link price α changes, both the optimum and the equilibria change. The optimum is examined first. The social costs of two candidate networks on four players are computed as follows:

- **Star**: 3 links. Sum of distances: center $1 + 1 + 1 = 3$, each leaf $1 + 2 + 2 = 5$; total $3 + 3 \cdot 5 = 18$. Social cost $3\alpha + 18$.
- **Complete graph**: 6 links. Everyone is at distance 1 from the others: $4 \cdot 3 = 12$. Social cost $6\alpha + 12$.

The two lines cross at $3\alpha + 18 = 6\alpha + 12$, that is, at $\alpha = 2$ (Figure 8.26). For $\alpha < 2$ the complete graph is cheaper; for $\alpha > 2$ the star is. The reason is clear: the complete graph has 3 more links than the star (cost 3α), and these links reduce the sum of distances by 6 (each brings two leaves 2 steps closer to each other). The extra links pay off when $3\alpha < 6$, that is, $\alpha < 2$. For general n too, the optimum is the complete graph for $\alpha \le 2$ and the star for $\alpha \ge 2$ (Fabrikant et al., 2003).

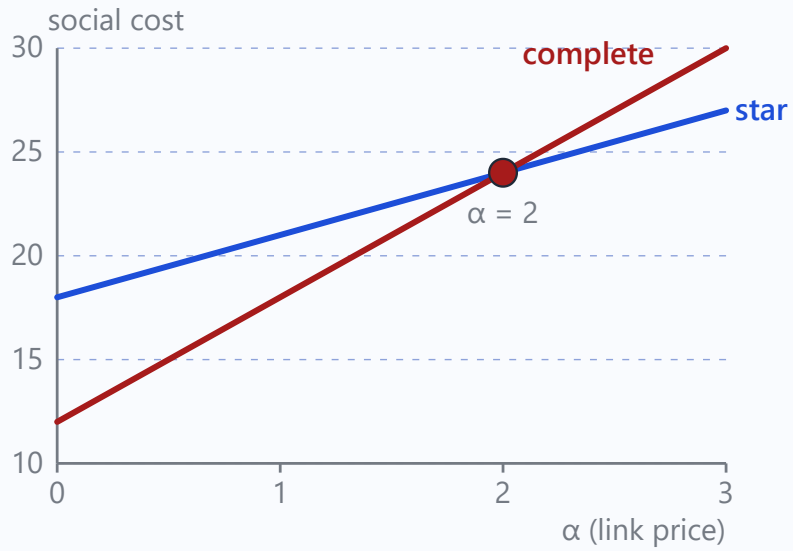


Figure 8.26: The social cost in the four-player network as a function of the link price α. The red line is the complete graph ($6\alpha + 12$), the blue line the star ($3\alpha + 18$); they cross at $\alpha = 2$.

α	Optimum	Equilibria
0.5	complete (15)	complete (15)
1.5	complete (21)	cycle 22, star 22.5
2.5	star (25.5)	star 25.5 · path 27.5

Table 8.14: Optimum and equilibria in the four-player network formation game (social costs; equilibria by brute-force scan).

Table 8.14 shows three regions.

- ***α < 1*: links are cheap.** A link buys at least 1 unit of distance and costs less than 1. So everyone connects directly: the complete graph is both an equilibrium and the optimum. The star, however, is not an equilibrium: at $\alpha = 0.5$ a leaf that buys a link to another leaf pays 0.5 and gains 1.
- ***1 < α < 2*: in between.** The optimum is still the complete graph, but the complete graph is no longer an equilibrium: a player who drops a link saves α and loses only 1. The equilibria at $\alpha = 1.5$ (star 22.5, cycle 22) cost more than the optimum (21).
- ***α > 2*: links are expensive.** The optimum is the star, and the star is also an equilibrium. But more expensive equilibria can appear alongside it. At $\alpha = 2.5$ the path network ($3 \cdot 2.5 + 20 = 27.5$) is also an equilibrium. This requires that the links be bought by the two middle players: had a player at an end bought his or her own link, that player could have reduced the distance by moving the link to a node closer to the middle.

A general result: as long as $\alpha \geq 1$, the star is always an equilibrium. If the center drops a link the network is disconnected; if a leaf buys a new link it gains at most 1 in distance and pays at least 1. But the equilibrium is often not unique, and some equilibria are markedly worse than the optimum. This loss ratio of selfish behavior is again studied under the name price of anarchy in network formation games.

8.5.7 Common mistakes in network formation

Common mistakes

- **Charging the cost of a link to both sides:** In the one-sided model only the buyer pays for a link. In the two-sided (Jackson–Wolinsky) example the cost is charged separately to each end. The models must not be mixed up.
- **Counting distances once in the social cost:** Each player sums his or her own distances; so in the social cost the distance of every pair is counted twice (in the complete graph there are 6 pairs and the sum of distances is 12).
- **Taking the distance to be infinite once a link is dropped:** When a link is dropped in the complete graph, the distance to that neighbor only rises to 2; there are other paths. A network becomes disconnected only if the sole path is cut.
- **Trying only adding links:** Deviations include adding a link, dropping one, and swapping (dropping one and buying another).
- **Looking at the gain of only one end in pairwise stability:** To form a link both ends must gain (or at least not lose).
- **Assuming the optimum is an equilibrium:** At $\alpha = 1.5$ the optimum (the complete graph) is not an equilibrium.

8.6 Control and incentives

The common conclusion of the previous sections is that the equilibrium reached by selfish users can differ from what is best for society. There are three ways to deal with this gap: gather the decisions in a center, change the users' incentives, or share the resources by a fairness criterion.

8.6.1 Centralized or distributed?

Who makes the routing decisions in a network is a basic design choice. Table 8.15 compares the two extreme approaches.

	Centralized routing	Distributed (selfish) routing
Who decides?	A traffic control center, an SDN (software-defined networking) controller, an Internet service provider (ISP) doing traffic engineering	Every driver / every router (navigation apps, BGP: the inter-domain routing protocol)
Goal	Smallest total latency $C(f)$ (Wardrop's second principle)	Smallest own latency (Wardrop's first principle)
Two-route example	1,500 / 1,500 → 71,250 car-min	2,000 / 1,000 → 75,000 car-min
Requires	All demand and latency information; compliance by the users	Only local information; no compliance problem
Weakness	At the optimum some are on the slower route → they want to deviate; single point of failure	Inefficient equilibria such as Braess's

Table 8.15: Centralized and distributed (selfish) routing.

There are examples of centralized routing in real networks. In **software-defined networking** (SDN) the routers do not decide on their own; a controller program sees the state of the whole network and tells each router what to do. An **Internet service provider** also does traffic engineering inside its own network, distributing traffic evenly over its lines. In these cases the center owns its network, and the packets it routes do not object.

The most important example of distributed routing is the Internet itself. The Internet is formed by the union of thousands of independent networks, and routes between these networks are chosen with **BGP** (Border Gateway Protocol, the inter-domain routing protocol). Each network chooses its routes according to its own commercial interests. In city traffic too, each driver follows his or her own navigation app.

The strengths and weaknesses of the two approaches complement each other. Centralized routing can aim at the optimum, but it needs all the demand and latency information and depends on the users complying. At the optimum some users are on the slower route (in the two-route example those on Route 2 experience 27.5 minutes while those on Route 1 experience 20); these users want to

deviate. Moreover, if the center fails the whole system is affected. Distributed routing works with local information only and has no compliance problem, because everyone is doing what he or she wants anyway; but it can reach inefficient equilibria, as in Braess's paradox.

Hence *intermediate solutions* are common in networks. The center does not control everything; it influences only part of it:

- **Price:** The center does not tell users what to do, but makes some routes more expensive (a congestion toll).
- **Information:** The center recommends routes; the user is free to follow the recommendation or not.
- **Partial control:** The center routes part of the traffic itself and the users choose the rest. This is called **Stackelberg routing**. The name comes from Stackelberg games, in which a leader moves first and the followers then respond to it: the center places its own traffic so as to improve the equilibrium reached by the selfish users.

8.6.2 Incentives, load balancing and applications

The right incentives steer users toward using network resources in a balanced way and raise efficiency. In traffic, tolls or warnings reduce congestion. In smart power grids, time-of-use tariffs balance the load at peak hours: if electricity is expensive in the evening, the washing machine is run at night. The cleanest example of incentives is the congestion toll mentioned in 8.5.1.

The idea is this: the reason the Wardrop equilibrium deviates from the optimum is that the driver does not see the delay $x \cdot \ell'(x)$ imposed on others. If each route is charged a toll equal to this externality at the optimum, the driver now looks at "time + toll". This sum equals the marginal cost of that route. When the drivers equalize these sums, the optimum, where the planner equalizes marginal costs, is reached.

Example 8.16

In the two-route network ($\ell_1 = 5 + x_1/100$, $\ell_2 = 20 + x_2/200$, 3,000 cars) let the tolls be measured in units of time (the equivalent in minutes).

1. The optimum is 1,500 / 1,500. The externalities at the optimum: on Route 1, $x_1 \cdot \ell_1' = 1{,}500 / 100 = 15$; on Route 2, $x_2 \cdot \ell_2' = 1{,}500 / 200 = 7.5$.
2. A toll worth 15 minutes is put on Route 1 and one worth 7.5 minutes on Route 2. The driver now compares $5 + x_1/100 + 15$ with $20 + x_2/200 + 7.5$.
3. The new equilibrium: $20 + x_1/100 = 27.5 + (3{,}000 - x_1)/200$. Multiply both sides by 200: $4{,}000 + 2x_1 = 5{,}500 + 3{,}000 - x_1$, that is, $3x_1 = 4{,}500$ and $x_1 = 1{,}500$.
4. At the tolled equilibrium the traffic splits exactly as at the optimum. Every driver's "time + toll" is 35 ($20 + 15 = 27.5 + 7.5$). The total travel time falls to 71,250 car-minutes.

> The tolls collected do not vanish from society; they can be channeled into road building or public transport. But from the driver's own viewpoint, time plus toll (35) is higher than the old equilibrium time (25). This is one reason why road pricing is politically difficult.

Wrong incentives, on the other hand, backfire. Strategic, selfish behavior increases and the result is congestion and low efficiency. Two examples are common. First, a navigation app that recommends the same “fastest” route to everyone can fill up side roads: the app sends each driver to the fastest road individually, but when everyone goes to the same road, that road slows down. Second, if a toll is chosen badly, traffic shifts to another bottleneck: when a bridge becomes tolled, all the vehicles may head for a free but narrow side bridge.

The concepts of this chapter are used in three major application areas:

- **The Internet.** ISPs route data packets; inter-domain route selection is done with BGP. Load balancing (ECMP) and traffic engineering reduce congestion.
- **Traffic management.** Route optimization, smart signaling and navigation apps. Traffic planners evaluate the effect of a new road by computing the Wardrop equilibrium the drivers will reach.
- **Energy.** Energy distribution and load shifting in smart power grids. When consumers use electricity forms a structure similar to a congestion game: if everyone uses it at the same hour the grid is strained.

8.6.3 Efficiency versus fairness: sharing bandwidth

Minimizing the total latency is not the only criterion. When the resources of a network are shared among users, *fairness* matters too. A small example shows this. Two links each have capacity 1. Flow A uses both; flow B uses only link 1; flow C uses only link 2. Each flow is given a rate (x_A, x_B, x_C). The capacity constraints are:

$$x_A + x_B \le 1, x_A + x_C \le 1$$

Which allocation should be chosen? Three criteria give three different answers (Table 8.16).

Criterion	What is optimized?	(x_A, x_B, x_C)	Total
Largest total flow	Σx_i	(0, 1, 1)	2 (A starves)
Max-min fairness (egalitarian criterion)	First the smallest share is made as large as possible, then the next	(1/2, 1/2, 1/2)	1.5
Proportional fairness (Nash social welfare)	The product of the shares is maximized (Kelly, 1997)	(1/3, 2/3, 2/3)	≈ 1.67

Table 8.16: Three criteria for sharing bandwidth.

Example 8.17

1. **Largest total flow.** A uses capacity on both links; every unit given to it takes one unit from B and one from C. In terms of the total, it is best to give A nothing: $x_A = 0$, $x_B = x_C = 1$, total 2. But A starves completely.
2. **Max-min fairness.** All shares are raised together at the same rate. When the shares reach 1/2, link 1 ($x_A + x_B = 1$) and link 2 ($x_A + x_C = 1$) fill up at the same time. Now no share can be increased without decreasing someone else's. The result is (1/2, 1/2, 1/2), total 1.5.
3. **Proportional fairness.** The product of the shares $x_A \cdot x_B \cdot x_C$ is maximized. To maximize the product no capacity is wasted: $x_B = x_C = 1 - x_A$. The product becomes $x_A (1 - x_A)^2$.
4. Maximizing the product is the same as maximizing its logarithm, because the logarithm is an increasing function: $\log x_A + 2 \log(1 - x_A)$. The derivative of $\log x$ is $1/x$. Setting the derivative to zero: $1/x_A - 2/(1 - x_A) = 0$, that is, $1/x_A = 2/(1 - x_A)$.
5. By cross-multiplication $1 - x_A = 2x_A$, $x_A = 1/3$. The allocation is (1/3, 2/3, 2/3), total 5/3 ≈ 1.67.

The difference between the three criteria is a value judgment. The largest-total-flow criterion cares about the total and is willing to shut one user out completely. Max-min fairness cares about the worst-off user; it equalizes everyone but sacrifices from the total. **Proportional fairness** stands between the two. A, which uses both links, gets less, because it consumes shares of two resources; but nobody falls to zero, because a zero share makes the product zero. The total (≈ 1.67) is higher than that of max-min and lower than the largest total flow.

It is no coincidence that proportional fairness is the same thing as Nash social welfare: both concepts ask for maximizing the product of the shares. The logarithmic form shows that this criterion means "raising any user's share by 10% is worth the same, whoever the user is". In 1998 Frank Kelly and coauthors interpreted the congestion control of the Internet (TCP-like protocols) from this angle (Kelly et al., 1998). Each link announces a *congestion price* that grows as the load on it grows; each flow adjusts its rate by looking at the prices along its own path. This distributed algorithm converges to the proportionally fair allocation without any center.

8.6.4 Discussion questions

The concepts of this chapter shed light on three questions that have no definite answer but are worth thinking about.

When navigation apps recommend the "fastest" route to everyone, does traffic approach the Wardrop equilibrium or the system optimum? Why? If the app recommends to each driver the fastest route under the current latencies, each driver finds his or her own best response more easily. This speeds up convergence to the Wardrop equilibrium, not to the system optimum. If the app

deliberately sent some drivers to slower routes, the optimum could be approached; but those drivers might not follow the recommendation. If a single app has very many users, it can act as a kind of Stackelberg leader.

A city is planning a new bridge. How can the equilibrium the drivers will reach once the bridge opens be computed in advance, to foresee a Braess effect? First the network is modeled: a latency function (for example BPR) for each road and the demands between source–destination pairs are determined. Then the Wardrop equilibrium is computed in the two networks, with and without the bridge; this means minimizing the Beckmann potential with a method such as Frank–Wolfe. The total latencies of the two equilibria are compared. If the total at the equilibrium with the bridge is higher, the bridge may do harm through selfish use; the remedy may be tolls or access restrictions.

Are routers on the Internet (BGP) selfish agents? Whose cost do they minimize? Individual routers follow a program; it is the networks that operate them that are selfish. Each network (an ISP, for example) chooses its routes according to its own commercial interests: it prefers networks that pay it, or with which it peers for free, over networks it has to pay. This is why Internet routes are often not the shortest or the fastest. What BGP minimizes is not the packet's delay but the network operator's cost.

8.7 Summary

- **Latency depends on load.** The latency of each edge is a nondecreasing function of the flow on it: constant, linear, BPR or queueing ($1/(c - x)$). Classical shortest-path algorithms (Dijkstra, Bellman–Ford) assume fixed weights; load-dependent latency turns routing into a game.
- **In selfish routing** each user minimizes his or her own latency. The latency of a path is the sum of the edge latencies; the total latency is $C(f) = \Sigma f_e \ell_e(f_e)$.
- **At a Wardrop equilibrium** all used paths have equal and lowest latency. In the two-route example (3,000 cars) the equilibrium is 2,000 / 1,000, everyone takes 25 minutes, and the total is 75,000 car-minutes.
- **At the system optimum** the marginal costs ($\ell + x \cdot \ell'$) are equalized: 1,500 / 1,500, total 71,250. The optimum is not an equilibrium; the gap comes from the externality $x \cdot \ell'$, which the driver does not take into account.
- **The Beckmann potential** $\Phi = \Sigma \int \ell_e$ is minimized at the Wardrop equilibrium; an equilibrium always exists and the edge latencies are unique. Bisection (`find_eq`) finds the equilibrium of a two-route network by halving the interval in each round.
- **Braess's paradox**: in the 4,000-car network a shortcut of 0 minutes raises everyone's time from 65 to 80 minutes; the zigzag is dominant for everyone. The optimum of the network with the shortcut is ≈ 64.7. The paradox does not occur in series-parallel networks; choosing which roads to close is hard in general networks.

- **In congestion games** Rosenthal's potential $\Phi = \Sigma_e \Sigma_{k \le n_e} c_e(k)$ changes by exactly the cost difference of the single player who deviates. Hence a pure Nash equilibrium exists and best-response dynamics stop (26 → 23 → 21 → 20); but in general they may not stop quickly (PLS-complete).
- **In network formation** there is a tension between link cost and reach. In the two-sided model the star and the cycle are pairwise stable; in the one-sided model at $\alpha = 1.5$ the star and the cycle are equilibria, and the optimum, the complete graph, is not. The optimum is the complete graph for $\alpha \le 2$ and the star for $\alpha \ge 2$.
- **Control and incentives**: centralized routing aims at the optimum but needs information and compliance; intermediate solutions (congestion tolls, route recommendations, Stackelberg routing) are common. In sharing bandwidth, largest total flow, max-min fairness and proportional fairness choose different allocations.

Concept	Essence	Example
Selfish routing	Everyone minimizes his or her own latency	3,000 cars, two routes
Wardrop equilibrium	Used routes are equal and fastest	2,000 / 1,000, 25 min
System optimum	Marginal costs are equal	1,500 / 1,500, 71,250 car-min
Braess's paradox	New road → everyone slower	65 → 80 min
Rosenthal's potential	$\Delta\Phi$ = the mover's cost difference	Φ: 26 → 23 → 21 → 20
Network formation	Link cost ↔ reach	$\alpha = 1.5$: star / cycle are equilibria, complete graph is the optimum

Table 8.17: The concepts of the chapter at a glance.

Equilibrium and optimum often come apart: the driver looks at his or her own latency, the planner at the marginal cost.

8.8 Key terms

network design routing node link edge graph capacity bandwidth router
star / ring / tree / mesh topology OSPF (Open Shortest Path First) equal-cost multipath (ECMP)
latency function BPR latency function M/M/1 queue selfish routing nonatomic atomic
Wardrop equilibrium user equilibrium system optimum marginal cost Beckmann potential
Braess's paradox series-parallel network Wheatstone network congestion game
Rosenthal potential potential function potential game finite improvement property PLS-complete
externality Metcalfe's law network formation game pairwise stability peering complete graph
social cost software-defined networking (SDN) Border Gateway Protocol (BGP)

Internet service provider (ISP) Stackelberg routing max-min fairness proportional fairness

8.9 Test Yourself

1. 5,000 cars go from A to B. Route 1: $\ell_1 = 10 + x_1/100$, Route 2: $\ell_2 = 20 + x_2/400$ (minutes). What are the Wardrop equilibrium, the latency and the total latency?

   Answer

   The extreme cases are not equilibria: if everyone is on Route 1, $\ell_1 = 60 > 20$; if everyone is on Route 2, $\ell_2 = 32.5 > 10$. Equality: $10 + x_1/100 = 20 + (5{,}000 - x_1)/400$. Multiplying by 400, $4{,}000 + 4x_1 = 8{,}000 + 5{,}000 - x_1$, $5x_1 = 9{,}000$, $x_1 = 1{,}800$, $x_2 = 3{,}200$. The latencies are $10 + 18 = 28$ and $20 + 8 = 28$ minutes. The total is $5{,}000 \times 28 = 140{,}000$ car-minutes.

2. What are the system optimum and its total latency in the same network? How many car-minutes of extra delay does selfish routing produce?

   Answer

   Marginal costs: Route 1 $10 + x_1/50$, Route 2 $20 + x_2/200$. Equality: $10 + x_1/50 = 20 + (5{,}000 - x_1)/200$; multiplying by 200, $2{,}000 + 4x_1 = 4{,}000 + 5{,}000 - x_1$, $5x_1 = 7{,}000$, $x_1 = 1{,}400$, $x_2 = 3{,}600$. The latencies are 24 and 29 minutes. The total is $1{,}400 \times 24 + 3{,}600 \times 29 = 33{,}600 + 104{,}400 = 138{,}000$. The difference is $140{,}000 - 138{,}000 = 2{,}000$ car-minutes. At the optimum those on Route 2 are 5 minutes slower and want to move to Route 1.

3. Let the Braess network have 3,000 cars; the narrow roads take $x/100$ and the wide roads 50 minutes. What is the equilibrium time before and after the shortcut (A→B, 0 minutes) opens? Does the paradox occur?

   Answer

   Before: 1,500 / 1,500, each route $15 + 50 = 65$ minutes. After: suppose everyone is on the zigzag; 3,000 cars on the narrow roads (30 minutes), the zigzag takes $30 + 0 + 30 = 60$. A driver who goes back to an old route experiences $30 + 50 = 80$; nobody goes back. The equilibrium is 60 minutes. The shortcut speeds everyone up by 5 minutes; the paradox does not occur. Because the most crowded narrow road (30) is much faster than the wide road (50), the shortcut is genuinely useful. The paradox appears only in a certain range of traffic.

4. In the atomic Braess network (s→A and B→t: $c_e(k) = k$; s→B and A→t: 4; A→B: 0), 3 drivers are in the profile (upper, lower, zigzag). What are Φ and the costs? How does Φ change if the driver on

the upper route moves to the zigzag?

Answer

User counts: s→A 2 (upper, zigzag), A→t 1, s→B 1, B→t 2 (lower, zigzag), A→B 1. $\Phi = (1 + 2) + 4 + 4 + (1 + 2) + 0 = 14$. Costs: upper $2 + 4 = 6$, lower $4 + 2 = 6$, zigzag $2 + 0 + 2 = 4$. If the upper driver moves to the zigzag, he or she stays on s→A (2), A→B 0, and there are 3 people on B→t (3): the new cost is $2 + 0 + 3 = 5$. The gain is 1; therefore Φ falls from 14 to 13. Check: 4 is deleted from A→t, 0 is added to A→B and 3 to B→t; $-4 + 3 = -1$. The only pure equilibrium of this game is the profile in which all three drivers are on the zigzag (6 each, total 18); the profile with the smallest total cost is (upper, lower, zigzag) (16).

5. In the one-sided network formation game let $n = 5$ players and $\alpha = 3$. What is the social cost of the star and of the complete graph? Which is the optimum? Is the star a Nash equilibrium?

Answer

Star: 4 links, cost $4 \cdot 3 = 12$. Distances: center $4 \cdot 1 = 4$, each leaf $1 + 3 \cdot 2 = 7$; total $4 + 4 \cdot 7 = 32$. Social cost 44. Complete graph: 10 links ($5 \cdot 4/2$), cost 30; sum of distances $5 \cdot 4 = 20$; social cost 50. Since $\alpha = 3 > 2$, the optimum is the star. The star is an equilibrium: a leaf gains only 1 in distance with a new link and pays 3; if the center drops a link the network is disconnected.

6. In the two-sided four-player example (benefit: 4 for distance 1, 2.2 for distance 2, 1 for distance 3) let the cost of a link to each end be c. For which range of c is the complete graph pairwise stable? The star?

Answer

Complete graph: a player who cuts a link saves c; the distance to that neighbor rises from 1 to 2 and he or she loses $4 - 2.2 = 1.8$. For stability $c \le 1.8$ is needed; there is no link to add. Star: if two leaves form a link each gains 1.8 and pays c; for it not to be formed, $c \ge 1.8$. If the center cuts a link it loses that leaf completely (4) and saves c; $c \le 4$ is needed. If a leaf cuts its link it loses 8.4. The star is stable for $1.8 \le c \le 4$. At $c = 1.8$ both are stable.

7. In the two-route network ($\ell_1 = 5 + x_1/100$, $\ell_2 = 20 + x_2/200$, 3,000 cars) a toll τ (the equivalent in minutes) is to be put on Route 1 only. For which value of τ does the tolled equilibrium coincide

with the system optimum?

Answer

The optimum is 1,500 / 1,500; there $\ell_1 = 20$, $\ell_2 = 27.5$. For the tolled equilibrium to be at 1,500 the driver must see the two routes as equal: $20 + \tau = 27.5$, that is, $\tau = 7.5$. With a smaller toll Route 1 is still too crowded; with a larger one it is too empty. This single toll gives the same toll difference (7.5) as putting separate marginal tolls (15 and 7.5) on the two routes; the driver's decision depends only on the difference.

8. If `find_eq(l1, l2, 3000)` is called with `eps = 10`, how many rounds does the loop run and what is the width of the final interval?

Answer

Each round halves the width: after k rounds the width is $3{,}000/2^k$. The loop runs as long as the width is greater than 10. For $2^8 = 256$ the width is $\approx 11.7 > 10$; for $2^9 = 512$ it is $\approx 5.86 \le 10$. The loop runs 9 rounds; the final width is about 5.86 cars. Compared with `eps = 50` this is only 3 more rounds.

8.10 Further reading

- T. Roughgarden, *Selfish Routing and the Price of Anarchy* (2005): presents selfish routing, Wardrop equilibrium, Braess's paradox and the hardness of network design in a single book, with step-by-step proofs.
- N. Nisan, T. Roughgarden, É. Tardos and V. Vazirani (eds.), *Algorithmic Game Theory* (2007), Chapters 18–19: presents routing games and network formation games through the potential function method, from a computer science viewpoint.
- D. Easley and J. Kleinberg, *Networks, Crowds, and Markets* (2010), Chapters 8 and 17: explains traffic networks, Braess's paradox and network effects with little mathematics and many examples.
- A. R. Karlin and Y. Peres, *Game Theory, Alive* (2017): covers traffic routing and potential games at the student level, with exercises.

8.11 References

1. M. Beckmann, C. B. McGuire, C. B. Winsten, *Studies in the Economics of Transportation*, Yale University Press, 1956.
2. D. Braess, “Über ein Paradoxon aus der Verkehrsplanung”, *Unternehmensforschung* 12, 258–268, 1968.
3. J. E. Cohen, P. Horowitz, “Paradoxical behaviour of mechanical and electrical networks”, *Nature* 352, 699–701, 1991.

4. D. Easley, J. Kleinberg, *Networks, Crowds, and Markets*, Cambridge University Press, 2010 (Chapter 8: traffic networks and Braess; Chapter 17: network effects).
5. A. Fabrikant, A. Luthra, E. Maneva, C. Papadimitriou, S. Shenker, "On a network creation game", *ACM PODC*, 2003.
6. A. Fabrikant, C. Papadimitriou, K. Talwar, "The complexity of pure Nash equilibria", *ACM STOC*, 2004.
7. M. Jackson, A. Wolinsky, "A strategic model of social and economic networks", *Journal of Economic Theory* 71(1), 44–74, 1996.
8. A. R. Karlin, Y. Peres, *Game Theory, Alive*, American Mathematical Society, 2017.
9. F. Kelly, "Charging and rate control for elastic traffic", *European Transactions on Telecommunications* 8, 33–37, 1997.
10. F. Kelly, A. Maulloo, D. Tan, "Rate control for communication networks: shadow prices, proportional fairness and stability", *Journal of the Operational Research Society* 49, 237–252, 1998.
11. W. Knödel, *Graphentheoretische Methoden und ihre Anwendungen*, Springer, 1969.
12. G. Kolata, "What if they closed 42d Street and nobody noticed?", *The New York Times*, December 25, 1990.
13. I. Milchtaich, "Network topology and the efficiency of equilibrium", *Games and Economic Behavior* 57, 321–346, 2006.
14. D. Monderer, L. Shapley, "Potential games", *Games and Economic Behavior* 14, 124–143, 1996.
15. N. Nisan, T. Roughgarden, É. Tardos, V. Vazirani (eds.), *Algorithmic Game Theory*, Cambridge University Press, 2007 (Chapters 18–19).
16. A. Rapoport, T. Kugler, S. Dugar, E. J. Gisches, "Choice of routes in congested traffic networks: Experimental tests of the Braess paradox", *Games and Economic Behavior* 65(2), 538–571, 2009.
17. R. W. Rosenthal, "A class of games possessing pure-strategy Nash equilibria", *International Journal of Game Theory* 2, 65–67, 1973.
18. T. Roughgarden, *Selfish Routing and the Price of Anarchy*, MIT Press, 2005.
19. T. Roughgarden, "On the severity of Braess's paradox: Designing networks for selfish users is hard", *Journal of Computer and System Sciences* 72, 922–953, 2006.
20. J. G. Wardrop, "Some theoretical aspects of road traffic research", *Proceedings of the Institution of Civil Engineers*, Part II, 1, 325–378, 1952.

PART III · CHAPTER 9

The Price of Anarchy

In traffic every driver chooses his or her own route, in a server cluster every job chooses its own machine, in a network every user rents his or her own link. None of them thinks about the total cost of the system. How much worse can the outcome of these selfish choices be than the best arrangement a central planner could build? The Price of Anarchy measures this loss with a single ratio; the Price of Stability asks the same question from the optimistic side. Starting from their definitions, these notes go step by step through Pigou's network, the 4/3 bound for linear latency and the smoothness method, the Braess network, the load balancing game, the H_n bound for fair cost sharing, and the congestion tolls that reduce the loss.

LEARNING OUTCOMES

Defining
Define the Price of Anarchy (PoA) and the Price of Stability (PoS); show the difference between them with an example.

Computing
Find the equilibrium and optimum costs in the Pigou, Braess and load balancing examples and compute the ratio.

Explaining the bound
Explain the intuitive proof of the bound PoA ≤ 4/3 for linear latency and why it is tight.

Improving
Show how a congestion toll turns the optimum into an equilibrium, in the two-route network and in Pigou's network.

9.1 Basic concepts

Nash equilibrium, the basic equilibrium concept of game theory, is a definition of *stability*: at equilibrium nobody wants to change his or her choice alone. The definition does not say that the equilibrium is a good outcome. In the Prisoner's Dilemma both players defect and end up worse off

than if they had both stayed silent. This chapter turns the question “How bad can an equilibrium be?” into a number. The question first becomes concrete in a small game, and the measure is defined afterwards.

9.1.1 A first game: load balancing

A server cluster has three servers: A, B and C. The servers have different speeds: A has speed 2, B and C have speed 1. Jobs arrive one after another and each job is placed on one of the servers. The **completion time** of a job is the total load of the server it is placed on divided by the server's speed. For example, if A holds 4 units of work, every job on A finishes in *4 / 2 = 2* time units. If B holds 2 units, the jobs on B finish in *2 / 1 = 2*, and if C holds 1 unit, the job on C finishes in *1 / 1 = 1*.

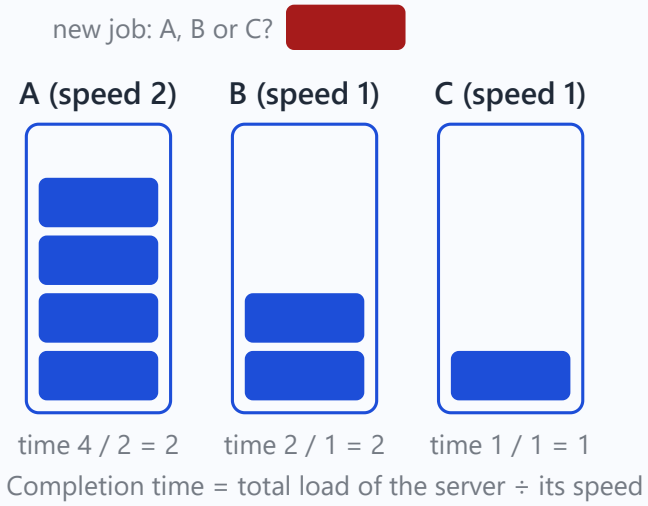


Figure 9.1: A moment in the load balancing game. A holds 4 units of work, B holds 2 and C holds 1; the completion times are 4/2 = 2, 2/1 = 2 and 1/1 = 1 respectively. The newly arrived job will choose one of the three servers.

In Figure 9.1 a new job has arrived. The job thinks only of its own completion time. If it goes to A, A's load becomes 5 and the job finishes in *5/2 = 2.5*. If it goes to B, it finishes in *3/1 = 3*; if it goes to C, in *2/1 = 2*. The selfish job picks C. This choice is best for that job but may not be best for the total cost of the system. The game asks two questions:

- If every job minimizes its own completion time, is the total cost of the system also minimized?
- If not, how much worse than the best assignment can the equilibrium reached by selfish jobs be, at most?

The answer to the first question is generally “no”. The answer to the second is a number that varies from game to game; this number is the real subject of the chapter. The same game is solved in detail in 9.4.8: the total time of the worst equilibrium turns out to be 13 and that of the best assignment 12.5, and the ratio is 1.04.

9.1.2 The Price of Anarchy: the bill for selfishness

The **Price of Anarchy** (PoA) measures the efficiency loss between the worst equilibrium produced by selfish behavior and the social optimum. The two concepts in this definition are:

- The **social optimum** is the best outcome with respect to the common goal of the system. In cost problems the social optimum is the state that *minimizes the total cost*. In traffic this is the distribution in which the total travel time of all drivers is smallest.
- An **equilibrium** is a state in which everyone chooses by looking at his or her own cost and nobody wants to change the choice alone.

A traffic network shows the tension between the two concepts well. Every driver tries to reduce his or her own travel time. But a driver entering a road slows everyone on that road a little, and the driver does not take this effect into account. As a result, while everyone acts sensibly from his or her own point of view, the total delay can be larger than necessary.

The history of the concept is short. In 1999 Elias Koutsoupias and Christos Papadimitriou, studying selfish players who distribute jobs over parallel machines, defined the ratio of the worst equilibrium to the optimum under the name “coordination ratio” (Koutsoupias and Papadimitriou, 1999). The name “price of anarchy” was given by Papadimitriou in 2001 (Papadimitriou, 2001). The name describes a comparison: the price of everybody deciding for himself or herself, that is, of anarchy, instead of a central “dictator” who decides everything.

Two numerical examples come up often in this chapter. The first is a two-route network: 3,000 cars split between two routes; the total delay is 75,000 car-min at the selfish equilibrium and 71,250 car-min in the best distribution. The ratio is $75{,}000 / 71{,}250 \approx 1.053$; that is, selfishness raises the total delay by about 5%. The second is the Braess network: when a new road is added to a network used by 4,000 cars, everyone's travel time rises from 65 minutes to 80 minutes. The Price of Anarchy measures such gaps with a single number and, more importantly, puts an upper bound on this number for a whole class of games.

9.1.3 Nash equilibrium and the social optimum

The ratio compares two different outcomes. Table 9.1 puts these two outcomes side by side.

	Nash equilibrium (individual)	Social optimum (system)
Definition	No player can improve his or her own outcome by changing strategy unilaterally.	Minimizes the total cost or maximizes the social welfare.
Represents	Individual incentives	The common goal of the system
How many?	There can be several (good and bad equilibria).	Its value is unique; several states can give the same value.
Stable?	Yes, by definition.	Often not: somebody wants to deviate.

Table 9.1: Nash equilibrium compared with the social optimum.

Let us briefly recall the definition of Nash equilibrium. There are n players. Each player i chooses a strategy s_i; the collection of all the choices $s = (s_1, \ldots, s_n)$ is a **strategy profile**. Player i's cost in this profile is written $C_i(s)$. s_{-i} denotes the strategies of the players other than i. A profile s is a **Nash equilibrium** if $C_i(s) \le C_i(s_i', s_{-i})$ holds for every player i and every alternative s_i'. In words: while the others stay put, no player can lower his or her cost by changing his or her own choice. In this chapter the players carry costs, not payoffs; that is why the inequality points in the "less than or equal" direction.

There are three typical reasons why an equilibrium differs from the optimum:

1. **Individual rationality.** Everybody looks at his or her own benefit. The total cost of the system appears in no player's objective function.
2. **Externalities.** One player's choice changes the cost of others, but the player does not take this effect into account. A driver entering a road slows the others; this slowdown does not appear on the driver's own bill. This cost, which is imposed on others and not taken into account by the decision maker, is called an **externality**.
3. **Lack of information.** Players may not see the whole system; even if they knew the best arrangement, there is no guarantee that they would follow it.

The Price of Anarchy takes the ratio of these two outcomes: the equilibrium is in the numerator and the optimum in the denominator.

9.1.4 Definitions: PoA and PoS

A game can have several Nash equilibria. Then "the cost of the equilibrium" is not a single number. There are two natural choices: look at the worst equilibrium, or look at the best one. The first gives the Price of Anarchy, the second the Price of Stability. The notation is as follows: NE is the set of Nash equilibria of the game; $C(s)$ is the total cost in state s; OPT is the state with the smallest total cost (the social optimum).

Definition 9.1

The **Price of Anarchy** of a cost game is the ratio of the cost of the worst Nash equilibrium to the cost of the optimum:

$$PoA = \max_{s \in NE} C(s) / C(OPT)$$

The **Price of Stability** is the ratio of the cost of the best Nash equilibrium to the cost of the optimum:

$$PoS = \min_{s \in NE} C(s) / C(OPT)$$

For the ratios to be defined, the optimum cost is assumed to be positive ($C(OPT) > 0$).

The two definitions differ in a single word: *max* or *min*. The PoA is a **pessimistic guarantee**: whichever equilibrium the players go to, the loss is at most this much. The PoS is an **optimistic scenario**: if somebody can steer the players to the best equilibrium, the loss drops to this much. The "stability" in the name of the PoS comes from the requirement that the outcome be stable (an equilibrium): if the optimum itself is not stable, how far from the optimum is the best of the stable outcomes?



Figure 9.2: The optimum and the equilibria on the cost axis. The optimum (OPT) is on the far left; the ratio of the best Nash equilibrium to the optimum gives the PoS, and the ratio of the worst Nash equilibrium to the optimum gives the PoA.

Figure 9.2 shows the definition on an axis. The cost increases to the right. By definition the optimum is on the far left, because no state can be cheaper than it. The equilibria lie to the right of the optimum or exactly on it. The leftmost equilibrium determines the PoS and the rightmost determines the PoA. The following result can be read directly from this picture:

Theorem 9.1 (Order of the ratios)

In every cost game that has an equilibrium and whose optimum cost is positive ($C(OPT) > 0$), $1 \le PoS \le PoA$. If the game has a single equilibrium, $PoS = PoA$.

Proof. Every equilibrium is a state, and the optimum is the cheapest of all states; hence $C(s) \geq C(OPT)$ for every equilibrium s. Since the costs are positive, both sides can be divided by $C(OPT)$, and the ratio of every equilibrium is at least 1; this includes the best equilibrium, so $PoS \geq 1$. The smallest element of a set cannot exceed its largest element; so $\min C(s) \leq \max C(s)$ and $PoS \leq PoA$. If there is a single equilibrium, the smallest and the largest element are the same element and the two ratios are equal.□

Example 9.1

Four jobs of sizes 2, 2, 1 and 1 are to be placed on two identical machines; the social cost is the maximum load. The equilibria of this game are found one by one in 9.4.4. Let us use the result here: the maximum loads of the equilibria are 3 and 4, and the optimum is 3.

1. **Worst equilibrium:** the equilibrium with maximum load 4. $PoA = 4 / 3 \approx 1.333$.
2. **Best equilibrium:** the equilibrium with maximum load 3. $PoS = 3 / 3 = 1$.
3. **Check:** $1 \leq 1 \leq 4/3$; the ordering theorem holds. PoS = 1 says that the optimum itself is also an equilibrium.

In payoff problems the ratio is inverted. In some games the players carry payoffs, not costs, and the goal of the system is to maximize the **social welfare** (SW for short). Then the worst equilibrium is the one with the smallest welfare. The ratio is written $SW(OPT) / SW(equilibrium)$ (the equilibrium welfare must be positive: $SW(equilibrium) > 0$); that is, the optimum moves to the numerator. The ratio is again at least 1, and the rule "a larger value means worse efficiency" is preserved. For example, if the optimum's welfare is 100 and the worst equilibrium's welfare is 80, the PoA is $100/80 = 1.25$.

9.1.5 Efficiency measures and how to read the ratio

The meaning of the ratio depends on how the social cost is defined. Two measures are used in this chapter:

- **Total cost**: the sum of the delay or resource usage costs of all users. This is the measure in the traffic examples; its unit is car-min (the minutes of every car are added up).
- **Makespan**: in a job assignment, the load of the most heavily loaded machine, that is, the time at which all jobs are finished. This is the classical measure used in load balancing.

In cost problems the PoA is the ratio of the worst equilibrium cost to the social optimum cost. A PoA of 1 shows that there is no efficiency loss: even the worst equilibrium is as good as the optimum. As the ratio grows, the loss caused by strategic behavior grows. The ratio converts easily to a percentage: if PoA = 1.25, the worst equilibrium is 25% more expensive than the optimum.

PoA	Meaning	Example
1	no loss	The Braess network, before the new road is opened

PoA	Meaning	Example
4/3	33% extra cost	Pigou's network; the bound for linear latency
2 − 2/(m + 1)	less than a factor of 2	load balancing, m identical machines
unbounded	no guarantee	steep latency (x^p)

Table 9.2: Reading PoA values. All the examples are worked out in detail in this chapter.

Why the worst case? At first sight looking at the worst equilibrium may seem too pessimistic. But this is exactly where the value of the PoA comes from. We often cannot know which equilibrium the players will go to: it depends on the starting state, the order of moves and chance. The PoA gives a guarantee that needs no such knowledge: "Whatever happens, the loss is at most this much." A network designer who knows that the PoA is small can be confident that the system will perform reasonably even without investing in central control. This idea is the game-theoretic counterpart of the approximation ratio in algorithms.

9.1.6 Two worst-case ratios: PoA and the approximation ratio

Algorithms courses study **approximation algorithms** for NP-hard problems. For these problems no method is known that finds the optimum in polynomial time; instead a fast algorithm looks for a good solution. The quality of the algorithm is measured by the **approximation ratio**: the worst-case ratio between the cost of the solution the algorithm finds and the cost of the optimum. The Price of Anarchy is also a worst-case ratio. The difference between the two lies in what prevents reaching the optimum.

Ratio	What prevents reaching the optimum	Numerator / denominator
Approximation ratio	computational power: a problem that cannot be solved in polynomial time (NP-hard)	the solution found by a fast algorithm ÷ optimum
Price of Anarchy	coordination: there is no central authority and everyone looks after his or her own interest	worst equilibrium ÷ optimum

Table 9.3: Two worst-case ratios. Both have the form "found / optimum"; what is missing is different.

The two ratios can be compared on the same problem. Distributing jobs over m identical machines to minimize the makespan is an NP-hard problem. The **greedy list scheduling algorithm** studied by Ronald Graham in 1966 takes the jobs in the given order and puts each job on the machine that is currently least loaded (Graham, 1966). The makespan this algorithm finds is at most $2 - 1/m$ times the optimum. If the jobs in the same problem are selfish players, the makespan of the pure Nash equilibria is at most $2 - 2/(m + 1)$ times the optimum (Finn and Horowitz, 1979). For $m = 2$ the two numbers are $3/2$ and $4/3$.

Example 9.2

Let there be two identical machines and three jobs of sizes 1, 1, 2 in this order. The list scheduling algorithm places the jobs in this order.

1. The first job of size 1: both machines are empty, so it goes to M1. Loads (1, 0).
2. The second job of size 1: the least loaded machine is M2. Loads (1, 1).
3. The job of size 2: both machines are equally loaded, so it goes to M1. Loads (3, 1); the makespan is 3.
4. The optimum: the job of size 2 alone on one machine and the two jobs of size 1 on the other: (2, 2); the makespan is 2.

The ratio is $3 / 2 = 2 - 1/2$; Graham's bound is attained exactly in this example. Moreover, the assignment (3, 1) is not a Nash equilibrium: if a job of size 1 on M1 moves to M2, it waits $1 + 1 = 2 < 3$ and gains. Selfish jobs do not stay in this bad assignment. The loss of the list scheduling algorithm comes from a computational restriction, and the loss of the selfish equilibrium from a lack of coordination.

The limits of the analysis. The PoA assumes players to be completely selfish and rational. Real players are not only selfish: trust, social norms, reputation and repeated interaction can increase cooperation. The PoA is therefore not a *prediction* but a pessimistic *guarantee*. The loss observed is often smaller than the bound the PoA gives; but it is known that the bound will not be exceeded.

9.1.7 Common mistakes in the definitions

Common mistakes

- **Confusing PoA with PoS:** The PoA looks at the worst equilibrium, the PoS at the best. “The ratio of an equilibrium to the optimum” is not always the PoA; only that of the worst equilibrium is.
- **Writing the ratio upside down:** In a cost problem the optimum is in the denominator; the ratio is never below 1. If a value like 0.75 was found, the ratio has been inverted.
- **Confusing a difference with a ratio:** “Worst equilibrium − optimum” is a difference and depends on the unit (car-min, seconds). The PoA is a unitless ratio; this is what makes systems of different scales comparable.
- **Assuming PoS = 1 whenever there is a single equilibrium:** A single equilibrium can be bad. Pigou's network has a single equilibrium and $PoA = PoS = 4/3$.
- **Taking the PoA to be a prediction:** The PoA does not say what will happen in a particular example but what can happen in the worst case.

9.2 Pigou's network

The smallest example that shows the Price of Anarchy is the two-route network given by the English economist Arthur Cecil Pigou in 1920 (Pigou, 1920). The example is so simple that the whole calculation fits on a sheet of paper. Yet it captures exactly the worst case in networks with linear latency.

9.2.1 Setup: nonatomic traffic and Wardrop equilibrium

Traffic networks are modeled by a **graph**: the nodes are intersections and the edges (links) are roads. Each edge has a **latency function** $\ell_e(x)$: the time each driver using edge e spends when the traffic on it is x. Latency rises with traffic or stays constant; a busy road is slow. Drivers go from a source node s to a destination node t and each chooses a path (a sequence of consecutive edges). The latency of a path is the sum of the latencies of the edges on it.

In Pigou's network and in most traffic examples the number of drivers is very large, and the effect of a single driver is so small that it can be neglected. In this case the traffic is taken to be **nonatomic** (divisible): 1 unit of traffic is thought of as a "fluid" made of countless very small drivers and can be divided in any proportion. For example 0.3 of the traffic can be on one route and 0.7 on the other. When a single driver changes routes, the latencies of the routes do not change, because one driver's share is close to zero.

Definition 9.2

In nonatomic traffic, a flow distribution is a **Wardrop equilibrium** if the latency of every route carrying traffic is no greater than the latency of any available route. Equivalently: all used routes have equal latency and no unused route is shorter.

This definition is the counterpart of Nash equilibrium in nonatomic traffic. A driver on a slower route would gain by switching to a faster one; at equilibrium there is no such driver. The definition takes its name from the principle stated in 1952 by the British transport engineer John Glen Wardrop. Wardrop's second principle is the **system optimum**: the distribution in which the total latency is smallest. In this chapter the system optimum and the social optimum are the same thing.

The total latency is computed as follows. Let f_e be the traffic on edge e. Each driver on this edge spends $\ell_e(f_e)$, and there is f_e traffic on the edge; the contribution of the edge to the total cost is "traffic × latency":

$$C(f) = \sum_e \ell_e(f_e) \cdot f_e$$

9.2.2 The equilibrium in Pigou's network

In Pigou's network 1 unit of traffic goes from s to t and there are two routes. Let the share of traffic on the upper route be x and on the lower route $1 - x$.

- **Upper route:** latency $\ell(x) = x$. It is fast when empty and slows down as it fills; if everyone is on top the latency is 1. Think of a narrow city street.
- **Lower route:** the latency is always 1. However many cars enter, the time does not change. Think of a wide but long ring road.

To find the equilibrium, let us look through a driver's eyes. On the lower route the time is 1. On the upper route the time is x, and since x is a share it never exceeds 1. So the upper route is never slower than the lower one. If there is a driver on the lower route and $x < 1$, that driver switches to the upper route and gains. These switches continue until $x = 1$. At $x = 1$ the latency of the upper route is 1 and equals that of the lower route; now nobody can gain by switching.

Example 9.3

1. **Candidate: x = 1/2.** 1/2 min on top, 1 min on the bottom. A driver on the bottom who switches to the top waits about 1/2 min and gains. Not an equilibrium.
2. **Candidate: x = 0.9.** 0.9 min on top, 1 min on the bottom. Those on the bottom gain again. Not an equilibrium.
3. **Candidate: x = 1.** Everyone is on top, the time is 1 min. A driver on top who switched to the bottom would also wait 1 min; there is no gain. The lower route is empty but no shorter than the upper route. The Wardrop condition holds: an equilibrium.
4. **Cost:** $C(\text{equilibrium}) = 1 \cdot 1 + 0 \cdot 1 = 1$. Everyone waits 1 min.

That the equilibrium is unique also follows from this reasoning: in every distribution with $x < 1$ the drivers on the bottom leave.

9.2.3 The social optimum: minimizing total cost

A central planner looks at the question differently. The planner chooses x and wants to minimize the total latency. On the upper route there is x traffic, each waiting x min; on the lower route there is $1 - x$ traffic, each waiting 1 min:

$$C(x) = x \cdot x + (1 - x) \cdot 1 = x^2 - x + 1$$

The smallest value of this function is found with the help of the **derivative**. The derivative is the slope of a curve at each point. At the lowest point of a trough the slope is zero: to the left of the point the curve descends (negative slope) and to the right it rises (positive slope). The derivative rules needed

in this chapter are these: the derivative of x^k is $k \cdot x^{k-1}$ (for example the derivative of x^2 is $2x$); the derivative of $c \cdot x$ is c; the derivative of a constant is 0; the derivative of a sum is the sum of the derivatives. The derivative is written $C'(x)$.

Example 9.4

1. **Differentiate.** $C(x) = x^2 - x + 1$; the derivative of x^2 is $2x$, that of $-x$ is -1, that of 1 is 0. Result: $C'(x) = 2x - 1$.
2. **Set to zero.** $2x - 1 = 0 \Rightarrow x = 1/2$.
3. **Trough or peak?** For $x < 1/2$, $C'(x) < 0$ (the curve descends); for $x > 1/2$, $C'(x) > 0$ (the curve rises). The point is a trough.
4. **Compute the value.** $C(1/2) = 1/4 - 1/2 + 1 = 3/4$. The same result comes from summing "traffic × latency": $1/2 \cdot 1/2 = 1/4$ on top, $1/2 \cdot 1 = 1/2$ on the bottom; the total is $3/4$.
5. **Check the endpoints.** $C(0) = 1$ (everyone on the bottom) and $C(1) = 1$ (everyone on top). Both are larger than $3/4$ ✓.

At the optimum half of the traffic is "sacrificed" to the lower route. These drivers wait 1 min, while those on top wait only 1/2 min. The optimum is *not fair* and *not stable*: the drivers on the bottom want to switch to the top, because they would wait half a minute there. Without central coercion this distribution does not arise by itself.

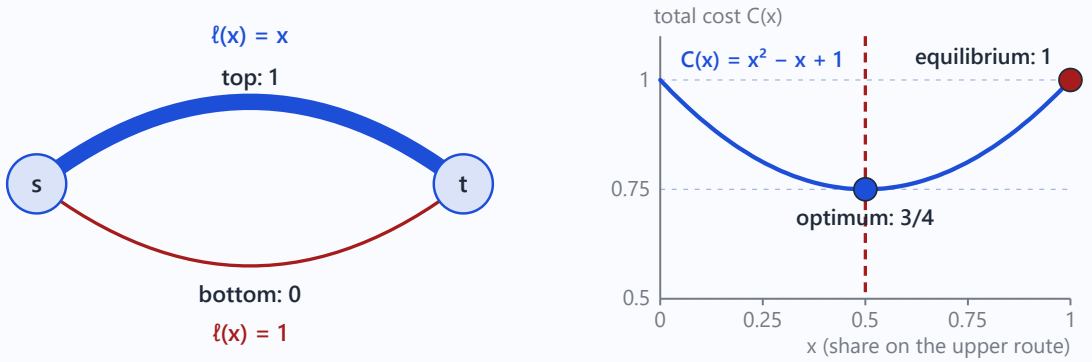


Figure 9.3: Pigou's network. On the left the network: the latency of the upper route is $\ell(x) = x$, that of the lower route 1; line thickness shows the traffic at equilibrium (everyone on top). On the right the total cost $C(x) = x^2 - x + 1$: the cost is 1 at the equilibrium $x = 1$ and 3/4 at the optimum $x = 1/2$.

Figure 9.3 shows the two results on the same graph. The right end of the parabola on the right is the equilibrium (red dot, cost 1); its lowest point is the optimum (blue dot, cost 3/4). The equilibrium sits at one of the worst points of the curve.

9.2.4 The ratio: PoA = 4/3

Pigou's network has a single equilibrium; so the worst equilibrium and the best equilibrium coincide, and the PoA and the PoS are equal:

$$PoA = C(\text{equilibrium}) / C(\text{optimum}) = 1 / (3/4) = 4/3 \approx 1.33$$

Selfish routing produces 33% more total delay than central planning. This number came out of a small example, but its importance is great: it has been proved that in *every* network with linear latencies, at *every* traffic volume, the ratio does not exceed 4/3 (9.3.4). Pigou's network is the worst example of this class.

9.2.5 Private cost and marginal cost

Why do the equilibrium and the optimum fall in different places? The answer is that the cost the driver looks at differs from the cost society looks at.

- **The driver looks at private cost.** On the upper route his or her own latency is x; the driver compares this with the 1 of the lower route. As long as $x < 1$ the driver picks the upper route. The result is $x = 1$.
- **Society looks at marginal cost.** The total cost of the upper route is $x \cdot x = x^2$. By how much does this total grow when a little traffic is added to the upper route? By its derivative: $2x$. The total cost of the lower route is $(1 - x) \cdot 1$, and every unit added there raises the total by exactly 1. The planner equates the two increases: $2x = 1 \Rightarrow x = 1/2$.

Definition 9.3

The **marginal cost** of an edge with latency function $\ell(f)$ carrying traffic f is the derivative of the edge's total cost $f \cdot \ell(f)$:

$$(f \cdot \ell(f))' = \ell(f) + f \cdot \ell'(f)$$

The first term is the added driver's own latency (the private cost). The second term $f \cdot \ell'(f)$ is the delay that driver adds to everyone else on the edge (the externality).

The formula in the definition comes from the product rule, but it can also be understood in words. When there are f cars on an edge, one more car enters. This car itself waits $\ell(f)$. In addition the latency of the edge rises by $\ell'(f)$, and this rise is passed on to each of the f cars on the edge; the total extra burden is $f \cdot \ell'(f)$.

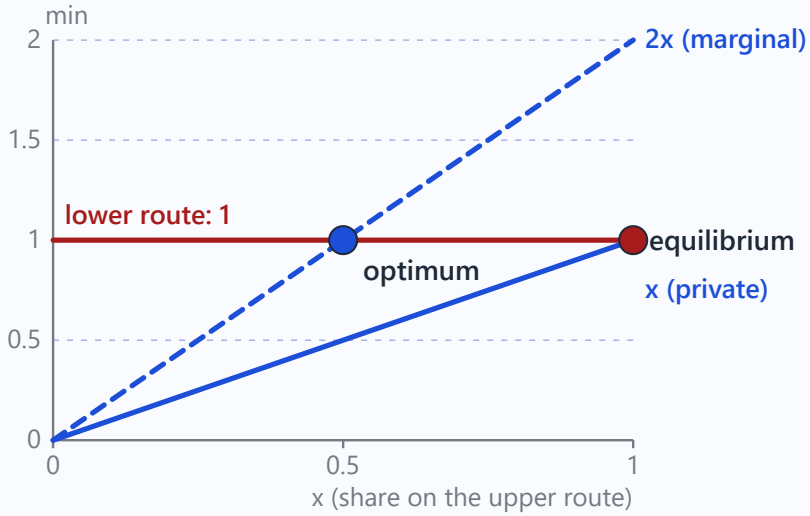


Figure 9.4: Private and marginal cost in Pigou's network. The solid blue line is the private cost x of the upper route, the dashed blue line its marginal cost $2x$, the red horizontal line the cost 1 of the lower route. The private cost meets 1 at $x = 1$ (equilibrium) and the marginal cost meets 1 at $x = 1/2$ (optimum).

Figure 9.4 shows the two crossings. Where the red line crosses the solid blue line is the equilibrium; where it crosses the dashed line is the optimum. The vertical distance between the two blue lines is $x \cdot \ell'(x) = x \cdot 1 = x$: the delay one driver imposes on the others. The driver does not see this difference; that is why more drivers than necessary enter the upper route.

This observation generalizes:

- **At equilibrium** the latencies of the used routes are equal.
- **At the optimum** the marginal costs of the used routes are equal; that is, “my own latency + the latency added to others because of me” is equal.

The difference between the two conditions is exactly the term $f \cdot \ell'(f)$. The congestion toll, the most direct tool for reducing the PoA, rests on this difference. If the driver is charged an amount equal to this externality, the cost the driver sees equals the marginal cost, and the selfish choice coincides with the optimum (9.6.2).

9.2.6 The two-route network: 3,000 cars

The same two conditions can be applied in a more realistic network. 3,000 cars split between two routes. Route 1 is short but narrow: $\ell_1 = 5 + x_1/100$ min. Route 2 is long but wide: $\ell_2 = 20 + x_2/200$ min. Here $x_1 + x_2 = 3{,}000$. Both latencies have the form $ax + b$, that is, they are linear: b is the time on the empty road and a is the slowdown each extra car brings.

Example 9.5

Equilibrium (equal latencies).

1. Substitute $x_2 = 3{,}000 - x_1$: $5 + x_1/100 = 20 + (3{,}000 - x_1)/200$.
2. To get rid of the fractions, multiply both sides by 200: $1{,}000 + 2x_1 = 4{,}000 + 3{,}000 - x_1$.
3. $3x_1 = 6{,}000 \Rightarrow x_1 = 2{,}000$, $x_2 = 1{,}000$.
4. Latencies: $\ell_1 = 5 + 20 = 25$, $\ell_2 = 20 + 5 = 25$ ✓. Everyone waits 25 min: $C(\text{equilibrium}) = 3{,}000 \cdot 25 = 75{,}000$ car-min.

Optimum (equal marginal costs). The marginal cost of Route 1 is $\ell_1 + x_1 \cdot \ell_1' = 5 + x_1/100 + x_1/100 = 5 + x_1/50$; that of Route 2 is $20 + x_2/100$.

1. $5 + x_1/50 = 20 + (3{,}000 - x_1)/100$. Multiply both sides by 100: $500 + 2x_1 = 2{,}000 + 3{,}000 - x_1$.
2. $3x_1 = 4{,}500 \Rightarrow x_1 = 1{,}500$, $x_2 = 1{,}500$.
3. Latencies: $\ell_1 = 5 + 15 = 20$, $\ell_2 = 20 + 7.5 = 27.5$. The optimum is again not fair: those on Route 2 wait 7.5 min longer.
4. Total: $1{,}500 \cdot 20 + 1{,}500 \cdot 27.5 = 30{,}000 + 41{,}250 = 71{,}250$ car-min.

Ratio: $PoA = 75{,}000 / 71{,}250 = 20/19 \approx 1.053$. Selfishness raises the total delay by about 5%; this is far below the 4/3 bound.

This example shows two things. First, in realistic networks the loss is often smaller than the worst-case bound. Second, in this example the optimal distribution forces some users (those on Route 2) to wait longer than at equilibrium. These drivers cannot be expected to follow the optimum voluntarily; an incentive such as a toll is needed to steer them. The toll computation for the same network is done in 9.6.1.

9.2.7 Common mistakes in Pigou's network

Common mistakes

- **Splitting the routes equally at equilibrium:** The thought "there are two routes, so the traffic splits half and half" is wrong. The equilibrium is where the latencies are equalized; in Pigou's network this is $x = 1$, with everyone on top. The half-and-half split is the optimum.
- **Thinking the driver looks at marginal cost:** The driver sees only his or her own latency. It is the planner, thinking about the total, who looks at marginal cost.
- **Summing only the latencies in the total cost:** Writing $C = \ell_1 + \ell_2$ is wrong. The latency of each route is multiplied by the traffic on that route: $C = x_1\ell_1 + x_2\ell_2$. At Pigou's equilibrium the lower route is empty and its contribution is 0, not 1.
- **Taking the optimum to be an equilibrium:** At $x = 1/2$ those on the bottom want to switch to the top; the optimum is not stable.

9.3 Linear latency and Braess

In Pigou's network the ratio came out as 4/3. Is this number a coincidence or part of a general law? The question splits into three subquestions. How does the ratio change when the latency function changes? Why does the ratio never exceed 4/3 in any network with linear latency? How does the same proof idea carry over to other games?

9.3.1 PoA grows as the latency gets steeper

In Pigou's network let us take the latency of the upper route to be $\ell(x) = x^p$ instead of $\ell(x) = x$; $p \geq 1$ is an integer. The latency of the lower route is again 1. As p grows the curve gets "steeper": while x is below 1, x^p is very small, and as x approaches 1 it suddenly jumps to 1. For example $0.5^1 = 0.5$, but $0.5^4 = 0.0625$. This kind of latency describes a road that behaves almost as if empty until it nears its capacity and then suddenly jams.

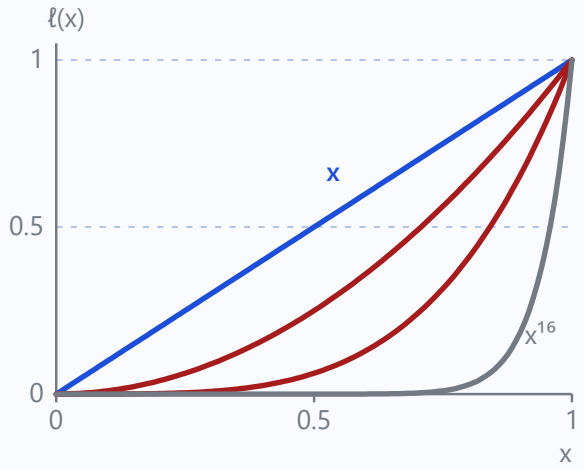


Figure 9.5: The latency of the upper route $\ell(x) = x^p$; the blue line is $p = 1$, the red curves are intermediate values, the gray curve is $p = 16$. All the curves reach the value 1 at $x = 1$, but as p grows they stay very low until that point.

The equilibrium does not change. For $0 \leq x \leq 1$, $x^p \leq 1$; the upper route is again never slower than the lower one. Everyone moves to the top and everyone waits 1 min: $C(\text{equilibrium}) = 1$.

The optimum improves. The total cost is $C(x) = x \cdot x^p + (1 - x) \cdot 1 = x^{p+1} + 1 - x$. To find the optimum the derivative is set to zero:

1. The derivative of x^{p+1} is $(p + 1)x^p$ and that of $1 - x$ is -1: $C'(x) = (p + 1)x^p - 1$.
2. $(p + 1)x^p = 1 \Rightarrow x^p = 1/(p + 1)$.
3. Take the p-th root of both sides: $x^{OPT} = (p + 1)^{-1/p}$. A negative exponent means "one over"; a fractional exponent $1/p$ is the p-th root. For example for $p = 2$, $3^{-1/2} = 1/\sqrt{3} \approx 0.577$.

4. The optimum cost simplifies: since $x^p = 1/(p+1)$, $x^{p+1} = x/(p+1)$ and $C(OPT) = x/(p+1) + 1 - x = 1 - x \cdot p/(p+1)$.

p	x^{OPT}	C(OPT)	PoA = 1 / C(OPT)
1	0.5	0.75	1.333
2	0.577	0.615	1.626
4	0.669	0.465	2.151
8	0.76	0.325	3.081
16	0.838	0.212	4.727

Table 9.4: The optimum share, the optimum cost and the PoA in Pigou's network when $\ell = x^p$. The equilibrium cost is 1 in every row.

Table 9.4 shows a trend. As the latency gets steeper the planner can put more traffic on the upper route, because the upper route is very fast until it nears its capacity. At the optimum the share on the upper route grows (from 0.5 to 0.838) and the optimum cost falls (from 0.75 to 0.212). The equilibrium stays where it is: everyone on top, everyone 1 min. The ratio therefore grows.

The growth is slow but unbounded. As p goes to infinity x^{OPT} approaches 1 and $C(OPT)$ falls to zero; as the denominator approaches zero the ratio grows without bound. Numerically, at $p = 10$ the PoA is about 3.51 and at $p = 100$ about 18.3. The ratio grows roughly at the rate $p / \ln p$. The lesson is this: **4/3 is not a universal constant**; it is specific to linear latency. What determines the bound is not the shape of the network but how “steep” the latency functions can be.

9.3.2 Interactive: latency exponent and PoA

Figure 9.6 does the same computation live. The slider changes the exponent p from 1 to 10. The graph on the left shows how the PoA grows with p; the bars on the right show how the traffic splits between the two routes at the equilibrium and at the optimum. Four numbers are written below: $x^* = (p+1)^{-1/p}$, $C(OPT)$, $C(equilibrium) = 1$ and the PoA.

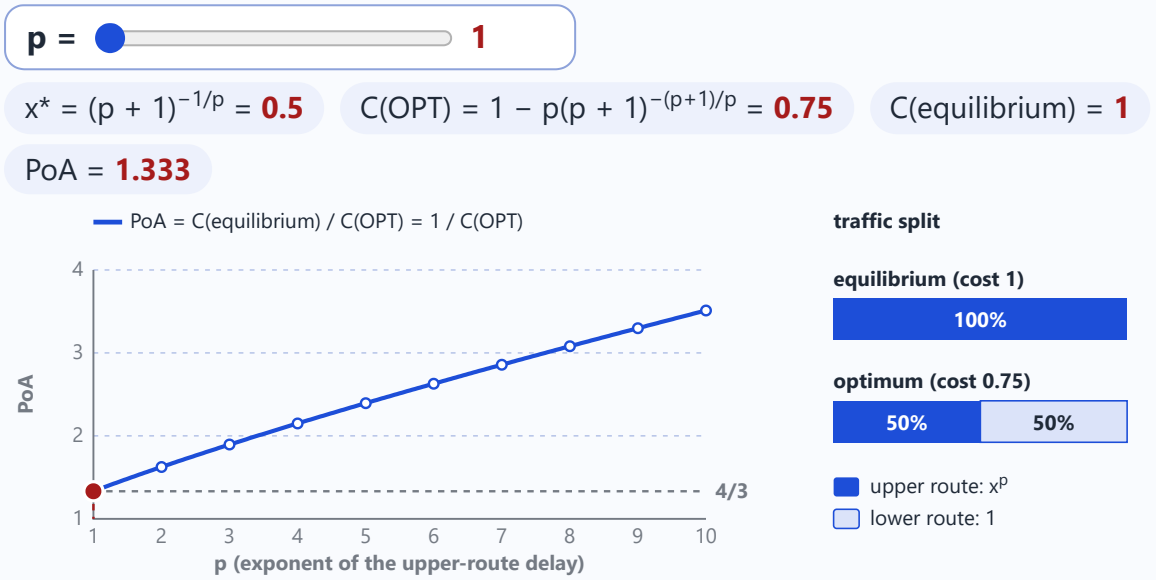


Figure 9.6: Interactive Pigou network. Upper route x^p, lower route 1; $C(x) = x^{p+1} + (1 - x)$, $C'(x) = 0 \Rightarrow x^* = (p + 1)^{-1/p}$, $PoA = 1 / C(x^*)$. The default value is $p = 1$: at the optimum 50% of the traffic is on top, $C(OPT) = 0.75$, PoA = 4/3.

In the graph the 4/3 level is marked with a dashed line. Only the point $p = 1$ falls exactly on this line; all the other points stay above it. As the slider is pulled to the right the equilibrium bar does not change (100% of the traffic on top), while the upper route's share in the optimum bar grows. The difference between the “rigidity” of the equilibrium and the “flexibility” of the optimum is the source of the growth of the ratio.

9.3.3 Code trace: pigou_poa(p)

The numbers in the table are produced by a short Python function. The function takes p and returns the PoA of Pigou's network.

```
def pigou_poa(p):          # top x**p, bottom 1
  x = (p + 1) ** (-1 / p)  # where C'(x) = 0
  opt = x ** (p + 1) + (1 - x)
  return 1 / opt           # eq. cost is 1

for p in [1, 2, 4, 8, 16]:
  print(p, round(pigou_poa(p), 3))
```

The code works line by line as follows:

- `x = (p + 1) ** (-1 / p)`: the point where the derivative is zero, that is, the upper-route share at the optimum. In Python `**` is the exponentiation operator; `-1 / p` is a fractional exponent.

- `opt = x ** (p + 1) + (1 - x)` : the total cost at this point. The first term is the contribution of the upper route ($x \cdot x^p$), the second that of the lower route.
- `return 1 / opt` : since the equilibrium cost is 1 for every p, the ratio is *1 / C(OPT)*.
- The last two lines call the function for $p = 1, 2, 4, 8, 16$ and print the result rounded to three digits.

p	$x = (p+1)^{-1/p}$	x^{p+1}	$1 - x$	opt	output
1	$2^{-1} = 0.5$	0.25	0.5	0.75	1,333
2	$3^{-1/2} \approx 0.5774$	0.1925	0.4226	0.6151	1,626
4	$5^{-1/4} \approx 0.6687$	0.1337	0.3313	0.4650	2,151
8	$9^{-1/8} \approx 0.7598$	0.0844	0.2402	0.3246	3,081
16	$17^{-1/16} \approx 0.8377$	0.0493	0.1623	0.2116	4,727

Table 9.5: Trace of `pigou_poa(p)`. Intermediate values are rounded to four digits.

The identity $x^{p+1} = x/(p + 1)$ can also be checked in each row. For example for $p = 4$, $0.6687 / 5 \approx 0.1337$ ✓.

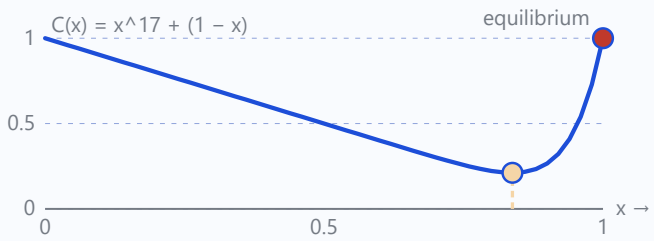


Figure 9.7: The total cost curve $C(x) = x^{17} + (1 - x)$ for $p = 16$. The curve falls for a long time like $1 - x$, reaches its minimum 0.212 at $x \approx 0.838$, and then rises quickly to the equilibrium ($x = 1$, cost 1). PoA = 4.727.

9.3.4 The 4/3 upper bound for linear latency

We saw that in Pigou's network the ratio is exactly 4/3. In 2002 Tim Roughgarden and Éva Tardos showed that this is no coincidence: in no network with linear latency does the ratio exceed 4/3 (Roughgarden and Tardos, 2002).

Theorem 9.2 (Roughgarden–Tardos, 2002)

If the latencies are linear, that is, $\ell_e(x) = a_e x + b_e$ with $a_e, b_e \geq 0$ for every edge, then in every network and for every traffic demand, the equilibrium of nonatomic traffic satisfies

$$C(\text{equilibrium}) \leq 4/3 \cdot C(\text{optimum})$$

The bound is tight: equality holds in Pigou's network.

The proof has three steps, and each step rests on a small idea. The notation is as follows: f_e is the traffic on edge e at equilibrium and f_e^{OPT} the traffic at the optimum. The total latency is the sum over edges of "traffic × latency": $C(f) = \Sigma_e \ell_e(f_e) \cdot f_e$.

Step 1: the equilibrium condition. At equilibrium everyone uses the shortest path under the current latencies. Let us run a thought experiment: "freeze" the latencies at their equilibrium values and move the drivers to their optimum routes. With frozen latencies no driver can find a path shorter than his or her equilibrium path, because the equilibrium path is already the shortest. So the optimum's distribution, measured with the frozen latencies, cannot be cheaper than the equilibrium itself:

$$\Sigma_e \ell_e(f_e) \cdot f_e \le \Sigma_e \ell_e(f_e) \cdot f_e^{OPT}$$

The left side is the true cost $C(f)$ of the equilibrium. The right side is a thought experiment: the routes are as slow as at equilibrium, but the traffic is distributed as at the optimum. What remains is to bound this right side by the true cost $C(f^{OPT})$ of the optimum. Step 2 does this edge by edge.

Example 9.6

In Pigou's network the equilibrium is $f = (1, 0)$ (top, bottom) and the optimum is $f^{OPT} = (1/2, 1/2)$. The latencies at equilibrium: upper route $\ell(1) = 1$, lower route 1.

1. **Left side:** $1 \cdot 1 + 1 \cdot 0 = 1$, the cost of the equilibrium.
2. **Right side:** the frozen latencies (1 and 1) are multiplied by the traffic at the optimum: $1 \cdot 1/2 + 1 \cdot 1/2 = 1$.
3. $1 \le 1$ ✓. Since the latencies of the two routes are equal at equilibrium, how the traffic is distributed does not change the frozen cost.

9.3.5 The edge inequality and summation

Step 2: the edge inequality. For every edge with linear latency the following inequality holds:

$$\ell_e(f_e) \cdot f_e^{OPT} \le \ell_e(f_e^{OPT}) \cdot f_e^{OPT} + \tfrac{1}{4} \cdot \ell_e(f_e) \cdot f_e$$

In words: the cost in the thought experiment (left side) exceeds the optimum's own cost on that edge by at most a quarter of the equilibrium's cost on that edge.

Why is it true? Drop the edge index, write simply f and f^{OPT}, and substitute $\ell(x) = ax + b$. The expressions split into two groups:

- **The terms with b.** On the left there is $b \cdot f^{OPT}$, on the right $b \cdot f^{OPT} + \tfrac{1}{4} \cdot b \cdot f$. The extra term on the right cannot be negative ($b, f \ge 0$); the right side is not smaller.

- **The terms with a.** On the left there is $a \cdot f \cdot f^{OPT}$, on the right $a \cdot (f^{OPT})^2 + a \cdot f^2/4$. The inequality needed is $a \cdot f \cdot f^{OPT} \le a \cdot (f^{OPT})^2 + a \cdot f^2/4$. Collecting everything on the right gives $0 \le a \cdot ((f^{OPT})^2 - f \cdot f^{OPT} + f^2/4) = a \cdot (f^{OPT} - f/2)^2$. It suffices that a square is nonnegative and $a \ge 0$.

The perfect square in the last step can be checked by expansion: $(u - v)^2 = u^2 - 2uv + v^2$; taking $u = f^{OPT}$ and $v = f/2$ gives $2uv = f \cdot f^{OPT}$ and $v^2 = f^2/4$. Equality holds only when $f^{OPT} = f/2$ (or when $a = 0$ and $b \cdot f = 0$). The constant ¼ comes from the coefficient of this perfect square.

Example 9.7

Pigou's upper route ($\ell = x$, that is, $a = 1, b = 0$): $f = 1, f^{OPT} = ½$.

1. Left: $\ell(1) \cdot ½ = 1 \cdot ½ = ½$.
2. Right: $\ell(½) \cdot ½ + ¼ \cdot \ell(1) \cdot 1 = ½ \cdot ½ + ¼ = ½$.
3. The two sides are equal, because $f^{OPT} = f/2$. Pigou's network uses the inequality right at its limit; this is why 4/3 is tight.

An edge without equality ($a = 1, b = 0, f = 2, f^{OPT} = ½$): left $2 \cdot ½ = 1$; right $½ \cdot ½ + ¼ \cdot 2 \cdot 2 = ¼ + 1 = 5/4$. $1 \le 5/4$ ✓; the difference is $(½ - 1)^2 = ¼$.

Step 3: sum up. The inequality of Step 2 is summed over all edges. The sum of the left sides is the right side of Step 1. The sum of the right sides is $C(f^{OPT}) + ¼ \cdot C(f)$. Combined with Step 1:

$$C(f) \le \Sigma_e \ell_e(f_e) f_e^{OPT} \le C(f^{OPT}) + ¼ \cdot C(f)$$

Move ¼ · C(f) to the left: $¾ \cdot C(f) \le C(f^{OPT})$. Divide both sides by $¾ \cdot C(f^{OPT})$: $C(f) / C(f^{OPT}) \le 4/3$. □

Since equality holds in Pigou's network, the bound is **tight**: no number smaller than 4/3 can be written. In 2003 Roughgarden showed a stronger result: in a class of latency functions that includes the constant functions, the worst example is always a network with two parallel links like Pigou's (Roughgarden, 2003). So the complexity of the network does not enlarge the PoA; only the class of latency functions determines the PoA. For the linear class this is 4/3, and for polynomials such as x^p it is the values in Table 9.4.

9.3.6 Testing the bound numerically

What a bound says is best understood by comparing it with known examples. Table 9.6 brings together the examples of this chapter.

Example	Equilibrium	Optimum	Observed in this example	Theoretical upper bound
Pigou (x and 1)	1	3/4	4/3 (equal to the bound)	4/3

Example	Equilibrium	Optimum	Observed in this example	Theoretical upper bound
Braess (4,000 cars)	320,000	258,750	80 / 64.69 ≈ 1.237	4/3
Braess, normalized (x, 1, 1, x, 0)	2	3/2	4/3 (equal to the bound)	4/3
Two routes, 3,000 cars	75,000	71,250	1.053	4/3
20,000 random linear parallel networks	—	—	largest 1.33	4/3
Load balancing, 7 jobs, speeds 2, 1, 1 (total time)	13	12.5	1.04	5/2 (atomic players)

Table 9.6: Observed ratios and theoretical upper bounds.

A theoretical upper bound is the worst case over a class (for example over all networks with linear latencies). The ratio observed in a particular example is often smaller. In the random experiment in the table, a script computed the equilibrium and the optimum in 20,000 networks with 2–4 parallel links, randomly chosen latency coefficients a, b and a randomly chosen traffic demand. The ratio exceeded 4/3 in no example; in examples resembling Pigou's (one link almost constant, the other linear starting from zero) it came close to 4/3. The experiment repeated for the computations of this chapter also found the largest ratio to be about 1.333.

The example in the last row belongs to a different class. The players there are individual jobs, not nonatomic traffic; the bound for such games is 5/2 and is treated in 9.3.9.

9.3.7 A single recipe: smoothness

The 4/3 proof uses only two ingredients: the definition of equilibrium and an inequality that holds on every edge. In 2009 Roughgarden turned this structure into a recipe (Roughgarden, 2009). The recipe is called **smoothness**, and the PoA bounds of many games can be proved with the same pattern.

The thought experiment. Let s be a state (every player has a strategy) and s^{OPT} the optimum. While the others stay at s, what would player i pay if only he or she switched to the strategy in s^{OPT}? This is player i's **deviation cost**.

Notation. s_{-i}: the strategies of the players other than i. $C_i(s_i^{OPT}, s_{-i})$: the deviation cost of i. $C(s) = \Sigma_i C_i(s)$: the total cost.

Definition 9.4

Let $\lambda > 0$ and $0 \le \mu < 1$ be two constants. If for every s and s^{OPT}

$$\Sigma_i C_i(s_i^{OPT}, s_{-i}) \le \lambda \cdot C(s^{OPT}) + \mu \cdot C(s)$$

holds, the game is **(λ, μ)-smooth**.

In words: the sum of the deviation costs does not exceed λ times the cost of the optimum plus μ times the cost of s. The inequality is required for *every* state s, not only for equilibria. The condition is therefore strong, but it is exactly this generality that is useful in the proof. As λ and μ get smaller, the bound obtained improves.

The edge inequality in the 4/3 proof is exactly a smoothness condition. Its left side is "the cost of the optimum's traffic at the equilibrium's latencies"; in nonatomic traffic this is the sum of the deviation costs. Its right side is $1 \cdot C(f^{OPT}) + \frac{1}{4} \cdot C(f)$. So nonatomic routing with linear latencies is $(1, \frac{1}{4})$-smooth.

9.3.8 From smoothness to a PoA bound

Theorem 9.3 (The smoothness bound)

In a (λ, μ)-smooth cost game, every Nash equilibrium s satisfies $C(s) \le \lambda / (1 - \mu) \cdot C(s^{OPT})$; that is, $PoA \le \lambda / (1 - \mu)$.

Proof. **1. Equilibrium.** s is a Nash equilibrium: nobody gains by deviating alone. In particular no player gains by switching to his or her strategy in the optimum either: $C_i(s) \le C_i(s_i^{OPT}, s_{-i})$ for every i.

2. Sum. These inequalities are summed over all players. The sum of the left sides is the total cost: $C(s) \le \Sigma_i C_i(s_i^{OPT}, s_{-i})$.

3. Apply smoothness. By definition the right side is at most $\lambda \cdot C(s^{OPT}) + \mu \cdot C(s)$. Thus $C(s) \le \lambda \cdot C(s^{OPT}) + \mu \cdot C(s)$.

4. Rearrange. Move $\mu \cdot C(s)$ to the left: $(1 - \mu) \cdot C(s) \le \lambda \cdot C(s^{OPT})$. Since $\mu < 1$, $1 - \mu$ is positive and both sides can be divided by $(1 - \mu) \cdot C(s^{OPT})$: $C(s) / C(s^{OPT}) \le \lambda / (1 - \mu)$. Since s is an arbitrary equilibrium, the bound holds for the worst equilibrium too: $PoA \le \lambda / (1 - \mu)$. □

Familiar examples. The edge inequality of the 4/3 proof means $\lambda = 1$, $\mu = \frac{1}{4}$: $1 / (1 - \frac{1}{4}) = 1 / (3/4) = 4/3$. For atomic (indivisible-player) linear congestion games one can take $\lambda = 5/3$, $\mu = 1/3$: $(5/3) / (2/3) = 5/2$.

Example 9.8

A game is known to be $(1, \frac{1}{2})$-smooth. The bound is $1 / (1 - \frac{1}{2}) = 2$. With the same λ and $\mu = \frac{1}{4}$ the bound would be 4/3. The effect of μ is large: as μ approaches 1, $1 - \mu$ approaches zero and the bound explodes. That is why the definition requires $\mu < 1$; with $\mu = 1$ Step 4 would require dividing by zero and no bound would result.

Why does it matter? The proof uses only the condition “nobody deviates alone”, and only for a single deviation (the switch to the strategy in the optimum). This condition also holds on average in looser kinds of equilibrium. For example in mixed equilibria, where players choose probabilistically, Step 1 can be written with expected values and the same bound results. The bound that follows from smoothness and is carried over in this way to broad kinds of equilibrium is called the **robust Price of Anarchy**.

9.3.9 Atomic congestion games: PoA ≤ 5/2

In nonatomic traffic the effect of a single driver is neglected. In many systems, however, the number of players is small and each player loads the resource noticeably: a few large data transfers, a few server jobs. Such games are called **atomic** congestion games. The model is as follows:

- Each player carries one unit of load and chooses a set of edges (a path).
- The edge cost is linear: if x players use an edge, each pays $c_e(x) = a_e x + b_e$.
- z_e: the number of people using edge e at equilibrium; y_e: the number using it at the optimum.

The difference is at a single point. In nonatomic traffic a driver who changes routes does not change the latency of the new route. In an atomic game a player who alone switches to his or her optimum route finds at most $z_e + 1$ people on edge e: the z_e who are there at equilibrium and himself or herself. This “+1” is the delay the player adds himself or herself.

The key inequality. For an edge with $c(x) = x$ the sum of the deviation costs is at most $y(z + 1)$: each of the y players using the edge at the optimum pays at most $z + 1$. The optimum's cost on that edge is $y \cdot y = y^2$ and the equilibrium's is z^2. The smoothness inequality needed, for all integers $y, z \geq 0$, is:

$$y(z + 1) \leq 5/3 \cdot y^2 + 1/3 \cdot z^2$$

y (s^{OPT}) \ z (load at equilibrium)	0	1	2	3	4	5
0	**0**	1	4	9	16	25
1	2	**0**	**0**	2	6	12
2	14	9	6	5	6	9
3	36	28	22	18	16	16
4	68	57	48	41	36	33
5	110	96	84	74	66	60

Figure 9.8: The table of the key inequality. Rows are y (load at the optimum), columns z (load at equilibrium); a cell is *3 × (right side – left side)*, so that no fractions appear. There is no negative cell; at the orange 0s the two sides are equal.

Figure 9.8 tests the inequality at small values one by one. The cell value is $3 \cdot (5/3 \cdot y^2 + 1/3 \cdot z^2 - y(z + 1)) = 5y^2 + z^2 - 3y(z + 1)$. No cell is negative; a script tried all the values for $y, z \le 60$ and found no negative value. The orange 0s are the equality points: $(y, z) = (0, 0)$, $(1, 1)$ and $(1, 2)$.

Example 9.9

1. **(y, z) = (1, 2):** left $1 \cdot 3 = 3$; right $5/3 \cdot 1 + 1/3 \cdot 4 = 5/3 + 4/3 = 3$. Equality; cell value 0.
2. **(y, z) = (2, 3):** left $2 \cdot 4 = 8$; right $5/3 \cdot 4 + 1/3 \cdot 9 = 20/3 + 3 = 29/3 \approx 9.67$. It holds; the difference is $5/3$, cell value $3 \cdot 5/3 = 5$.
3. **(y, z) = (1, 0):** left $1 \cdot 1 = 1$; right $5/3$. It holds; cell value $5 - 3 = 2$.
4. **Conclusion:** the inequality holds on every edge; summing over the edges, the game is $(5/3, 1/3)$-smooth and $PoA \le (5/3)/(2/3) = 5/2$.

The same result holds for a general linear cost $a\,x + b$: the part with a is a times the inequality above, and the part with b holds by itself in the form $b \cdot y \le 5/3 \cdot b \cdot y$ (since a constant cost is unaffected by crowding). The 5/2 bound is also tight; this was shown independently by Awerbuch, Azar and Epstein and by Christodoulou and Koutsoupias in 2005 (Awerbuch et al., 2005; Christodoulou and Koutsoupias, 2005).

The comparison is instructive: the bound is 4/3 for nonatomic traffic and 5/2 for indivisible players. The source of the difference is the "+1" in $z + 1$: the delay the player adds himself or herself. At $(y, z) = (1, 1)$ the left side is 2; with a smaller λ, for example $(1, 1/3)$, the right side would be $1 + 1/3 = 4/3 < 2$ and the inequality would fail. So in games with large players the loss can be larger.

9.3.10 PoA in the Braess network

Braess's paradox is the most surprising result of selfish routing: adding a new road to a network can raise everyone's travel time. The German mathematician Dietrich Braess demonstrated this phenomenon in 1968. Here the paradox is viewed through the PoA.

In the network 4,000 cars go from s to t. There are four old links: $s \to A$ and $B \to t$ depend on traffic ($x/100$ min), $A \to t$ and $s \to B$ are constant (45 min). Before the new road opens there are two routes, and by symmetry the traffic splits evenly: 2,000 cars on each route, time $2{,}000/100 + 45 = 65$ min. This distribution is both the equilibrium and the optimum; PoA = 1.

Let a road with latency 0 ($A \to B$) open between A and B. A third route is born: $s \to A \to B \to t$ (the zigzag).

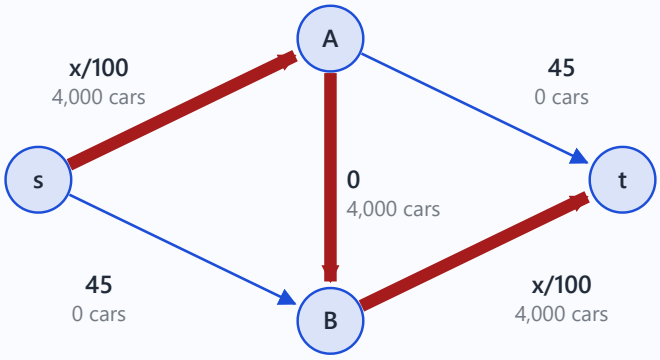


Figure 9.9: The equilibrium in the Braess network with the new road. All 4,000 cars use the zigzag route ($s \to A \to B \to t$, red); the time per person is $40 + 0 + 40 = 80$ min, the total 320,000 car-min.

The equilibrium. In Figure 9.9 everyone is on the zigzag. Let us check: there are 4,000 cars each on $s \to A$ and $B \to t$, each taking $4{,}000/100 = 40$ min. The zigzag takes $40 + 0 + 40 = 80$ min. A driver who switches to an old route waits $40 + 45 = 85$ min; since 85 > 80 nobody switches. The equilibrium cost is $4{,}000 \cdot 80 = 320{,}000$ car-min. The new road has raised everyone's time from 65 to 80 minutes.

A common comparison: 80/65 ≈ 1.23. This ratio compares the equilibrium of the new network with the equilibrium of the *old* network and measures the Braess loss. But the denominator of the PoA is the optimum of the *same network*. The question is: in the network with the new road, can the planner achieve an average better than 65 minutes?

Does the planner use the zigzag? Look at the marginal costs in the old distribution (2,000 on each old route, zigzag empty). There are 2,000 cars on $s \to A$ and the latency is 20 min. The marginal cost is $\ell + x \cdot \ell' = 20 + 2{,}000 \cdot (1/100) = 40$ min. The zigzag uses two such edges: $40 + 0 + 40 = 80$. The old route is $40 + 45 = 85$ (the marginal cost of a constant edge equals its own latency). $80 < 85$: even the planner sends a few cars onto the zigzag.

Example 9.10

By symmetry, assume that at the optimum there are a cars on each old route and $z = 4{,}000 - 2a$ cars on the zigzag.

1. **Edge loads.** Edge $s \to A$ is used by the a cars on the route $s \to A \to t$ and the z cars on the zigzag: $a + z = 4{,}000 - a$. By symmetry the same holds for $B \to t$.
2. **Total cost.** Each of the two variable edges contributes $(4{,}000 - a)$ cars × $(4{,}000 - a)/100$ min; each of the two constant edges contributes $a \cdot 45$; the new road contributes 0: $C(a) = 2(4{,}000 - a)^2/100 + 90a$.
3. **Derivative.** The derivative of $(4{,}000 - a)^2$ with respect to a is $-2(4{,}000 - a)$. $C'(a) = -4(4{,}000 - a)/100 + 90 = 0 \Rightarrow 4{,}000 - a = 2{,}250 \Rightarrow a = 1{,}750, z = 500$.
4. **Cost.** $C(1{,}750) = 2 \cdot 2{,}250^2/100 + 90 \cdot 1{,}750 = 101{,}250 + 157{,}500 = 258{,}750$ car-min. The average time is $258{,}750 / 4{,}000 \approx 64.69$ min; slightly better than 65.

5. **Check: are the marginal costs equal?** On $s \to A$ there are 2,250 cars, the latency is 22.5 min, the marginal cost $22.5 + 22.5 = 45$. Zigzag: $45 + 0 + 45 = 90$. Old route: $45 + 45 = 90$. Equal ✓. A script scanned all distributions, including nonsymmetric ones, and found the same optimum.
6. **Ratio.** $PoA = 320{,}000 / 258{,}750 \approx 1.237 \le 4/3$.

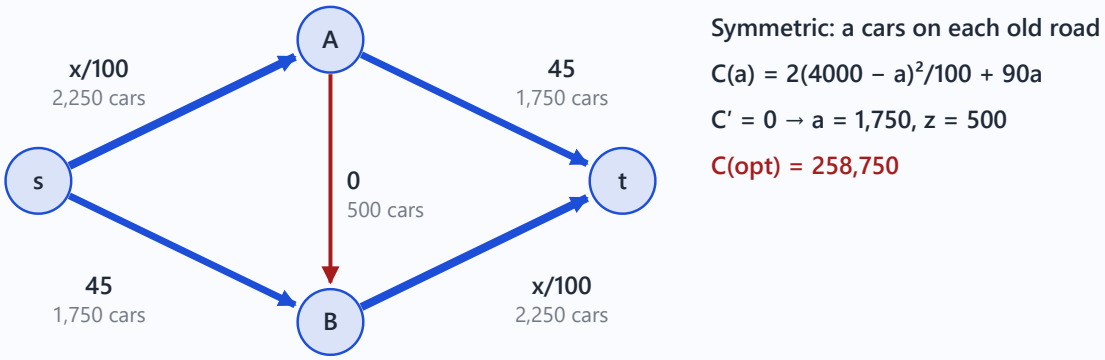


Figure 9.10: The optimum in the Braess network with the new road: 1,750 cars on each old route and 500 on the zigzag. *C(OPT) = 258,750* car-min.

Figure 9.10 shows the optimal distribution. The new road has improved the optimum slightly (from 260,000 to 258,750) but made the equilibrium much worse (from 260,000 to 320,000). The PoA measures these two effects within the same network.

The normalized Braess network. In textbooks the network is usually given with 1 unit of traffic and simple latencies: $s \to A$: x, $A \to t$: 1, $s \to B$: 1, $B \to t$: x, $A \to B$: 0. At equilibrium everyone is on the zigzag: $1 + 0 + 1 = 2$; a driver who switches to an old route waits $1 + 1 = 2$ and does not gain. The optimum is to close the zigzag and split the traffic half and half: each driver waits $1/2 + 1 = 3/2$. The ratio is $2 / (3/2) = 4/3$: this network also reaches the 4/3 bound exactly. In the 4,000-car network the times of the constant and variable edges are not so well matched; that is why the ratio stays below the bound, at 1.237.

9.3.11 Common mistakes with the bounds

Common mistakes

- **Taking 4/3 to be universal:** The bound is for linear latency and nonatomic traffic. With latency x^p the PoA grows without bound; for atomic players the bound is 5/2.
- **Assuming the ratio grows in complicated networks:** By Roughgarden's result the worst example always has two parallel links; the network structure does not enlarge the ratio.
- **Putting the old network in the denominator in Braess:** 80/65 is the Braess loss, not the PoA. The denominator of the PoA is the optimum of the same network (258,750).

- **Checking smoothness only at equilibrium:** The definition requires it for every state s; the proof uses this generality.
- **Taking μ to be 1 or larger:** If $1 - \mu \leq 0$ the division cannot be done or the sign flips; no bound results.

9.4 Load balancing

The Price of Anarchy was first defined not in traffic networks but in a model of distributing jobs over parallel machines (Koutsoupias and Papadimitriou, 1999). Today this model appears in server clusters, in task placement in the cloud and in distributing work over processor cores. It differs from traffic in two ways: the players are indivisible (each job goes to a single machine) and the social cost is often not the total but the maximum load.

9.4.1 The load balancing game

Definition 9.5

In the **load balancing game** there are n jobs (players) and m identical machines; the size of job j is w_j. Each job chooses a machine. The **load** of a machine is the sum of the sizes of the jobs placed on it. The cost of a job is the load of the machine it chose (all the jobs on that machine finish in a time equal to the machine's load). The **social cost** is the maximum load of the machines (the **makespan**), that is, the time at which all jobs finish.

In this model, if job j moves from machine k to machine h, its new cost is $load(h) + w_j$, because it adds its own size to the current load on h. If this number is smaller than the present cost $load(k)$, the move is profitable. An assignment is a Nash equilibrium if there is no profitable move for any job:

$$\text{for every } j \text{ and } h\text{: } load(h) + w_j \geq load(k_j)$$

Here k_j is the current machine of j. A “move” to its own machine is meaningless: for $h = k_j$ the left side is $load(k_j) + w_j$, which is always larger than the right side.

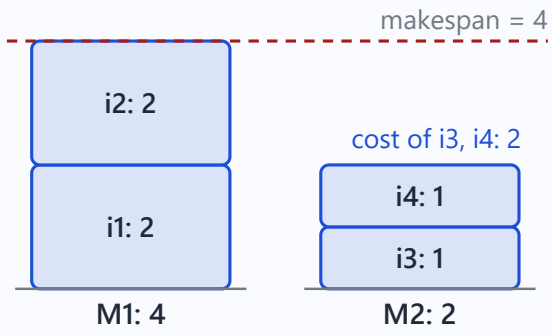


Figure 9.11: Four jobs on two identical machines: i1 and i2 of size 2 on M1 (load 4), i3 and i4 of size 1 on M2 (load 2). The makespan is 4; the cost of the jobs on M2 is 2.

In this game a pure Nash equilibrium always exists. This is not true in every game; in Rock–Paper–Scissors, for example, there is no pure equilibrium. In load balancing the following argument shows that an equilibrium exists. Start from any assignment. If the assignment is not an equilibrium there is a profitable move: job j moves from k to h and $load(h) + w_j < load(k)$. After the move the load of k falls and the load of h rises; but the new load of h is smaller than the old load of k. If the machine loads are sorted from largest to smallest into a list, this list "decreases" after the move: comparing the two lists from the left, at the first place where they differ the new list has a smaller number. In this comparison, which resembles dictionary order, the list cannot decrease forever, because the ways of distributing jobs over machines are finite in number and no list can repeat. So the profitable moves end somewhere; where they end is a Nash equilibrium.

Example 9.11

Let jobs of sizes 2, 2, 1, 1 be initially all on M1. The loads are (6, 0), the sorted list [6, 0].

1. i1 (size 2) moves to M2: $0 + 2 = 2 < 6$. The loads are (4, 2), the sorted list [4, 2]. The first element fell from 6 to 4: the list decreased.
2. i3 (size 1) moves to M2: $2 + 1 = 3 < 4$. The loads are (3, 3), the list [3, 3]. It decreased again.
3. Now no job can gain: a job of size 2 would wait $3 + 2 = 5$ on the other machine, a job of size 1 would wait $3 + 1 = 4$. The process stops at the equilibrium (3, 3).

9.4.2 Step by step: the {2, 2, 1, 1} example

Four jobs of sizes 2, 2, 1 and 1 are to be distributed over two identical machines. Each job wants to minimize its own completion time, that is, the total load of its machine. The social cost is the maximum load.

Example 9.12

1. **Social optimum.** The total work is *2 + 2 + 1 + 1 = 6*. Since there are two machines, the maximum load is at least *6/2 = 3*. The assignment {2, 1} | {2, 1} gives loads 3 and 3 and reaches this lower bound. OPT = 3.
2. **Another assignment.** The two large jobs on M1 (load 4), the two small jobs on M2 (load 2). The maximum load is 4. Can this be an equilibrium?
3. **A large job deviates.** If i1 (size 2) moves to M2 it waits *2 + 2 = 4*; it now waits 4. No gain. The same holds for i2.
4. **A small job deviates.** If i3 (size 1) moves to M1 it waits *4 + 1 = 5*; it now waits 2. Worse. The same for i4.
5. **Conclusion.** Nobody gains: {2, 2} | {1, 1} is a Nash equilibrium and its maximum load is 4.
6. **Ratios.** The optimum is also an equilibrium (at 3 | 3, a job of size 2 that moves waits 5 and a job of size 1 that moves waits 4). The worst equilibrium is 4, the best is 3: *PoA = 4/3*, *PoS = 1*.

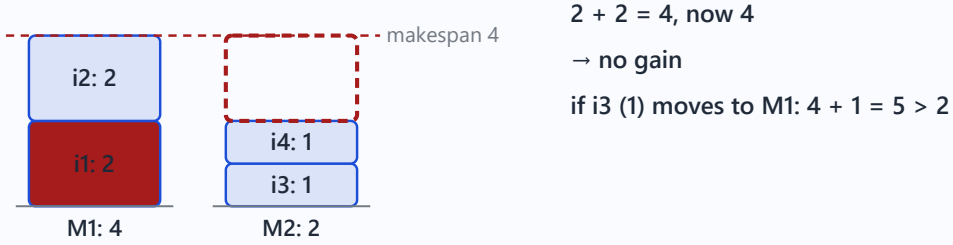


Figure 9.12: Deviation check in the assignment {2, 2} | {1, 1}. If i1 moves to M2 it waits *2 + 2 = 4*, and it already waits 4: no gain. If i3 moves to M1 it waits *4 + 1 = 5 > 2*.

Why the bad equilibrium arises in this example can be seen in Figure 9.12. M1 is overloaded, but the jobs there are large; if they move to M2 they wait 4 again because of their own size. The small jobs on M2 see no reason to go to M1. Nobody sacrifices himself or herself to "balance the load". With a tie (4 < 4 is false) the job stays where it is; this rule comes from the strict inequality in the definition.

9.4.3 Code trace: is_nash()

The function that checks whether an assignment is an equilibrium is as follows. `sizes` is the list of job sizes, `assign` is the machine index of each job (0 = M1, 1 = M2), and `m` is the number of machines.

```
def is_nash(sizes, assign, m):
  load = [0] * m
  for j, k in enumerate(assign):
    load[k] += sizes[j]
  for j, k in enumerate(assign):
    for h in range(m):
      if load[h] + sizes[j] < load[k]:
        return False      # j moves
  return True             # nobody moves
```

The function works in two stages:

- **Compute the loads.** `load = [0] * m` builds a list of `m` zeros. In the first loop `enumerate(assign)` gives each job's index `j` and its machine `k`; the size of the job is added to the load of that machine.
- **Try the deviations.** The second loop tests the condition `load[h] + sizes[j] < load[k]` for every job `j` and every machine `h`. If the condition is true, job `j` gains by moving to `h`, and the function returns `False` at once. If no deviation is profitable it returns `True`.

The trace of the call `is_nash([2, 2, 1, 1], (0, 0, 1, 1), 2)` is given in Table 9.7. For $h = k$ (its own machine) the condition is always false, because `load[k] + sizes[j] < load[k]` is impossible; the table shows only the steps with $h \neq k$.

Step	Variables	Operation	Result
1	—	`load = [0, 0]`	[0, 0]
2	j = 0, k = 0	i1 (size 2) to M1	[2, 0]
3	j = 1, k = 0	i2 (size 2) to M1	[4, 0]
4	j = 2, k = 1	i3 (size 1) to M2	[4, 1]
5	j = 3, k = 1	i4 (size 1) to M2	[4, 2]
6	j = 0, h = 1	2 + 2 < 4?	no
7	j = 1, h = 1	2 + 2 < 4?	no
8	j = 2, h = 0	4 + 1 < 2?	no
9	j = 3, h = 0	4 + 1 < 2?	no
10	—	loop finished	`True`

Table 9.7: Trace of `is_nash([2, 2, 1, 1], (0, 0, 1, 1), 2)`.

No job could move to a profitable machine; the function returns `True`: {2, 2} | {1, 1} is a Nash equilibrium. Had the same function been called with the assignment (0, 0, 0, 1), the loads would be [5, 1] and it would return `False` at the first deviation test (*1 + 2 = 3 < 5* for i1). The cost of the function is *O(nm)* comparisons for *n* jobs and *m* machines.

9.4.4 Brute force: all 16 assignments

Since each of the four jobs can choose one of two machines, there are $2^4 = 16$ assignments. Because the machines are identical, swapping M1 and M2 gives the same split; the 16 assignments therefore fall into five splits.

Split (M1 | M2)	Number of assignments	Maximum load	Nash?
{2, 1} | {2, 1}	4	3	✓ equilibrium
{2, 1, 1} | {2}	4	4	✗
{2, 2} | {1, 1}	2	4	✓ equilibrium
{2, 2, 1} | {1}	4	5	✗
{2, 2, 1, 1} | { }	2	6	✗

Table 9.8: All splits of the jobs {2, 2, 1, 1} over two machines (swapping M1 and M2 is counted in the same row).

The numbers of assignments are found as follows. For {2, 1} | {2, 1}, one of the jobs of size 2 (2 ways) and one of the jobs of size 1 (2 ways) go to M1: $2 \cdot 2 = 4$. For {2, 2} | {1, 1} the large jobs are either on M1 or on M2: 2. The total is $4 + 4 + 2 + 4 + 2 = 16$ ✓.

In each of the non-equilibrium splits there is a job that moves away. In {2, 1, 1} | {2}, if a job of size 1 on M1 moves to M2 it waits $2 + 1 = 3 < 4$. In {2, 2, 1} | {1}, if a job of size 2 moves it waits $1 + 2 = 3 < 5$. If all are on a single machine, any job gains by moving to the empty machine. The equilibria are those with maximum load 3 and 4: $PoA = 4/3$ (worst equilibrium), $PoS = 1$ (best equilibrium = optimum).

Best-response dynamics. This process, in which the players make profitable moves in turn, is called **best-response dynamics**. In this game the dynamics always stop at an equilibrium, because every profitable move shrinks the sorted load list (9.4.1). But the equilibrium reached depends on the starting point and the order of moves. The order in Example 9.11 reached (3, 3); if the large jobs start on M1 and the small ones on M2, the process never begins and stays at (4, 2).

9.4.5 Code trace: poa_pos()

Table 9.8 was built by hand. A function that scans all the assignments can do the same job; for each assignment the function calls `is_nash`.

```
from itertools import product
def poa_pos(sizes, m):
  opt, eqs = float("inf"), []
  for assign in product(range(m), repeat=len(sizes)):
    load = [0] * m
    for j, k in enumerate(assign):
      load[k] += sizes[j]
    opt = min(opt, max(load))
    if is_nash(sizes, assign, m):       # eq. test
      eqs.append(max(load))
  return max(eqs) / opt, min(eqs) / opt
r = poa_pos([2, 2, 1, 1], 2)
print(round(r[0], 3), r[1])
```

- `product(range(m), repeat=len(sizes))` produces in turn all the tuples giving each job a machine index: (0, 0, 0, 0), (0, 0, 0, 1), …, (1, 1, 1, 1). With four jobs and two machines there are 16 tuples.
- For each tuple the loads are computed and `opt` is updated to the smallest maximum load seen so far. `float("inf")` is an infinitely large initial value; at the first comparison it is certain to give way to a real number.
- If the assignment is an equilibrium, its maximum load is appended to the list `eqs`.
- At the end the ratios of the worst and the best equilibrium to the optimum are returned: `(PoA, PoS)`.

Assignment	Loads	Deviator	Equilibrium?	opt
(0, 0, 0, 0)	[6, 0]	i1: 0 + 2 = 2 < 6	no	6
(0, 0, 0, 1)	[5, 1]	i1: 1 + 2 = 3 < 5	no	5
(0, 0, 1, 0)	[5, 1]	i1: 1 + 2 = 3 < 5	no	5
(0, 0, 1, 1)	[4, 2]	none	yes, 4	4
(0, 1, 0, 0)	[4, 2]	i3: 2 + 1 = 3 < 4	no	4
(0, 1, 0, 1)	[3, 3]	none	yes, 3	3
(0, 1, 1, 0)	[3, 3]	none	yes, 3	3
(0, 1, 1, 1)	[2, 4]	i3: 2 + 1 = 3 < 4	no	3
(1, 0, 0, 0)	[4, 2]	i3: 2 + 1 = 3 < 4	no	3
(1, 0, 0, 1)	[3, 3]	none	yes, 3	3
(1, 0, 1, 0)	[3, 3]	none	yes, 3	3
(1, 0, 1, 1)	[2, 4]	i3: 2 + 1 = 3 < 4	no	3
(1, 1, 0, 0)	[2, 4]	none	yes, 4	3
(1, 1, 0, 1)	[1, 5]	i1: 1 + 2 = 3 < 5	no	3
(1, 1, 1, 0)	[1, 5]	i1: 1 + 2 = 3 < 5	no	3

Assignment	Loads	Deviator	Equilibrium?	opt
(1, 1, 1, 1)	[0, 6]	i1: 0 + 2 = 2 < 6	no	3

Table 9.9: Trace of `poa_pos([2, 2, 1, 1], 2)`. The "Deviator" column is the first profitable deviation that `is_nash` found.

When the scan ends, `eqs = [4, 3, 3, 3, 3, 4]` and `opt = 3`. The function returns *(4/3, 3/3) = (1.333..., 1.0)*; the last line prints this as `1.333 1.0`. The cost of the method is high: with n jobs and m machines there are m^n assignments. With 30 jobs and 2 machines this number exceeds a billion. Brute force is good for verifying small examples; for large games the PoA must be bounded by analysis.

9.4.6 The general result: 2 − 2/(m + 1)

In the {2, 2, 1, 1} example the PoA came out as 4/3. This is the worst possible value for two machines. The general result is as follows (Finn and Horowitz, 1979; Koutsoupias and Papadimitriou, 1999):

Theorem 9.4 (Pure PoA on identical machines)

On m identical machines, if the social cost is the maximum load, the Price of Anarchy of the pure Nash equilibria is

$$PoA = 2 - 2/(m + 1)$$

This value is tight.

Values: for $m = 1$ it is 1 (there is no choice with a single machine), for $m = 2$ it is 4/3, for $m = 3$ it is 3/2, for $m = 9$ it is 1.8. As m grows the ratio approaches 2 but never reaches 2. Selfish job distribution, whatever the number of machines, cannot be worse than twice the best distribution.

Why the bound stays below 2 can be seen by a short argument. Let L be the load of the most heavily loaded machine at an equilibrium. If that machine has a single job, L is the size of that job and the optimum must also put this job on some machine; the ratio is 1. If it has several jobs, let w be the size of the smallest job there. At equilibrium this job does not want to move to another machine: for every machine h, $load(h) + w \geq L$, that is, every machine has a load of at least $L - w$. The total amount of work is therefore at least $L + (m - 1)(L - w)$. The optimum cannot be smaller than the total work divided by the number of machines (the average load): $OPT \geq L - w(m - 1)/m$. Also, since the most heavily loaded machine has at least two jobs and the smallest is w, $w \leq L/2$. Putting the second fact into the first gives $OPT \geq L - (L/2)(m - 1)/m = L \cdot (m + 1)/(2m)$. Rearranging gives $L / OPT \leq 2m/(m + 1) = 2 - 2/(m + 1)$. Example 9.13 shows that this bound is attained, that is, that it is tight.

Example 9.13

Six jobs of sizes 3, 3, 3, 1, 1, 1 are to be distributed over three identical machines. The bound is $2 - 2/4 = 3/2$. This example reaches the bound.

1. **Optimum.** The total work is 12, with three machines: the maximum load is at least 4. {3, 1} | {3, 1} | {3, 1} achieves this: OPT = 4.
2. **The bad assignment.** {3, 3} | {3} | {1, 1, 1}: loads 6, 3, 3.
3. **A job of size 3 on M1:** if it moves to M2 it waits $3 + 3 = 6$, if it moves to M3 it waits $3 + 3 = 6$; it now waits 6. No gain.
4. **The job of size 3 on M2:** it now waits 3; if it moved to M3 it would wait 6, to M1 9. No gain.
5. **A job of size 1 on M3:** it now waits 3; if it moved to M2 it would wait 4, to M1 7. No gain.
6. **Conclusion.** The assignment is an equilibrium and the ratio is $6/4 = 3/2$. A script that scanned all $3^6 = 729$ assignments confirmed that the maximum loads of the equilibria are 4 and 6.

Mixed equilibria. If jobs can choose machines randomly (mixed strategies) the situation changes. Then the expected maximum load is used as the social cost, and the PoA can no longer be bounded by a constant: it grows with m very slowly but without bound. The growth rate is of the order $\log m / \log \log m$ (Koutsoupias et al., 2003; Czumaj and Vöcking, 2002). This resembles the "balls into bins" problem: when m balls are thrown at random into m bins, more balls than expected pile up in the fullest bin. Randomness leads to unlucky collisions even at equilibrium.

9.4.7 Machines with different speeds (related machines)

Real servers do not all run at the same speed. In the **related machines** model the speed of machine i is s_i and the completion time of a job is the total load of its machine divided by s_i. A fast machine finishes the same load in a shorter time. In this model too a pure Nash equilibrium always exists (the same "sorted list decreases" idea works with completion times).

Example 9.14

Speeds A = 2, B = 1, C = 1; job sizes 2, 1, 1. The social cost is the maximum completion time.

1. **Optimum.** A {2}, B {1}, C {1}: times $2/2 = 1$, $1/1 = 1$, $1/1 = 1$. The maximum time is 1. This is the best, because the job of size 2 takes 1 even on the fastest machine.
2. **A bad assignment.** A {1, 1}, B {2}, C { }: times $2/2 = 1$, $2/1 = 2$, 0. The maximum time is 2.
3. **The job of size 2 deviates.** It now waits 2. If it moves to A it waits $(2 + 2)/2 = 2$, if it moves to C $2/1 = 2$. No gain.

4. **A job of size 1 deviates.** It now waits 1. If it moves to C it waits *1/1 = 1*, if it moves to B *3/1 = 3*. No gain.
5. **Conclusion.** The second assignment is also an equilibrium. The ratio is *2/1 = 2*.

Assignment	Times (A, B, C)	Maximum	Nash?
A {2}, B {1}, C {1}	1, 1, 1	1	yes (optimum)
A {1, 1}, B {2}, C { }	1, 2, 0	2	yes

Table 9.10: Two equilibria on related machines (speeds A = 2, B = 1, C = 1; jobs 2, 1, 1).

The ratio is 2, which is greater than the bound $2 - 2/4 = 1.5$ for three identical machines. On identical machines the loss of a pure equilibrium always stays below a factor of 2. If the speeds differ, the loss grows without bound as the number of machines increases (though very slowly); the growth is of the order $\log m / \log \log m$ (Czumaj and Vöcking, 2002).

9.4.8 Back to the game: total completion time

The measure in the game at the beginning of the chapter is different. There the social cost is not the maximum load but **the sum of the completion times of all the jobs**. There are 7 jobs of size 1 and the server speeds are A = 2, B = 1, C = 1. If a server holds L jobs, each finishes in time L / s; the contribution of that server to the sum is $L \cdot L / s$.

Loads (A, B, C)	Total time	Nash?
(3, 2, 2)	$3 \cdot 1.5 + 2 \cdot 2 + 2 \cdot 2 = 12.5$	yes (optimum)
(4, 2, 1)	$4 \cdot 2 + 2 \cdot 2 + 1 \cdot 1 = 13$	yes (worst)

Table 9.11: The two equilibria of the game (7 units of work, speeds 2, 1, 1).

Example 9.15

1. **Is (4, 2, 1) an equilibrium?** A job on A waits 2; if it moves to C it waits *(1 + 1)/1 = 2*, if it moves to B it waits 3. No gain. A job on B waits 2; if it moves to C it waits 2, if it moves to A *5/2 = 2.5*. No gain. The job on C waits 1; that is already the best. Equilibrium ✓.
2. **Is (3, 2, 2) an equilibrium?** A job on A waits 1.5; if it moved to B or C it would wait 3. A job on B waits 2; if it moved to A it would wait *4/2 = 2* (equal, no gain), to C 3. Equilibrium ✓.
3. **Ratio.** *PoA = 13 / 12.5 = 1.04*. Its mirror image (4, 1, 2) is also an equilibrium with the same cost (13). A script that scanned all load distributions confirmed that the total times of the equilibria are only 12.5 and 13, and that the smallest total is 12.5.

Where does selfish placement lead? The jobs arrive one after another. What happens if each job, at the moment it arrives, picks the server on which its own completion time will be shortest (breaking ties in favor of the server that comes first alphabetically)? Table 9.12 follows this process step by step.

Job	If it goes to A	If it goes to B	If it goes to C	Choice	Loads
1	1/2 = 0.5	1	1	**A**	(1, 0, 0)
2	2/2 = 1	1	1	**A** (tie)	(2, 0, 0)
3	3/2 = 1.5	1	1	**B** (tie)	(2, 1, 0)
4	1.5	2	1	**C**	(2, 1, 1)
5	1.5	2	2	**A**	(3, 1, 1)
6	4/2 = 2	2	2	**A** (tie)	(4, 1, 1)
7	5/2 = 2.5	2	2	**B** (tie)	(4, 2, 1)

Table 9.12: Greedy selfish placement. In each row the completion time of the new job on the three servers; the chosen server is in bold.

The order is A, A, B, C, A, A, B, and the result is exactly (4, 2, 1), one of the worst equilibria. Greedy placement picks the best for the new job at each step but cannot foresee the effect of the jobs that arrive later; the sixth job going to A makes A expensive for the seventh job.

With 6 jobs and three identical servers the only equilibrium is (2, 2, 2) and it is the optimum: *PoA = 1*. The loss appears when the servers differ. One last comparison: the 4/3 in the {2, 2, 1, 1} example measures the maximum load, while this game measures the total time. **The value of the PoA depends on the social cost measured**; when one speaks of "the PoA of this game", it must always be stated which cost is meant.

9.4.9 Common mistakes in load balancing

Common mistakes

- **Not adding one's own size when deviating:** A job moving to h waits $load(h) + w_j$, not $load(h)$. In {2, 2} | {1, 1}, saying that a large job sees M2 with load 2 and "2 < 4, so it moves" is wrong; after moving it waits 4.
- **Moving on a tie:** The condition in the definition is strict (<). A move to an equal cost is not profitable; the job stays where it is.
- **Assuming the optimum is always an equilibrium:** In this example the optimum is an equilibrium (PoS = 1), but this is not so in every game; in Pigou's network the optimum is not an equilibrium.
- **Mixing up the measures:** The maximum load and the total time give different PoA values. The social cost must be stated explicitly before the calculation starts.
- **Carrying the bound for mixed equilibria over to pure ones:** $2 - 2/(m + 1)$ is for pure equilibria; for mixed equilibria the ratio grows slowly with m.

9.5 The Price of Stability

In most of the examples so far the PoA and the PoS were the same, or the PoS was 1. There is also a family of games in which the two ratios can be very different: network design games, in which players rent links and the cost is shared equally among those using them. In these games the worst equilibrium can be very bad, but there is always a good equilibrium too. The tool that shows how good that equilibrium is, is the potential function.

9.5.1 PoA ≠ PoS in the same game

Definition 9.6

In a network design game with **fair cost sharing** each link has a fixed cost c_e. Each player chooses a path between his or her own source and destination nodes. If n_e players use a link, each pays c_e / n_e. The total cost is the sum of the costs of the links used.

In this game there is an effect opposite to congestion: the more people use a link, the lower the cost per person. The players do not push one another; they attract. So it is good for the players to gather on the same link; the problem is which link they gather on.

Example 9.16

4 players rent a connection from *s* to *t*. There are two options: link A costs 2 TL, link B costs 6 TL. Those who use a link split its cost equally.

1. **Everyone on A.** Each pays *2/4 = 0.5* TL. A player who moved alone to B would pay 6 TL. Nobody moves: equilibrium. The total is 2 TL, which is also the optimum (the cheapest link is used).
2. **Everyone on B.** Each pays *6/4 = 1.5* TL. A player who moved alone to A would pay 2 TL; since *2 > 1.5* nobody moves. This is also an equilibrium; the total is 6 TL.
3. **A split profile (1 on A, 3 on B).** The player on A pays 2 TL, those on B pay *6/3 = 2* TL. If a player on B moves to A, two people share A and the player pays *2/2 = 1* TL; *1 < 2*, so the player leaves. Not an equilibrium.
4. **All profiles.** When all $2^4 = 16$ profiles are scanned, only "all on A" and "all on B" turn out to be equilibria. In the profile with 2 on A and 2 on B, a player on B pays 3 TL and would pay *2/3* TL on A; with 3 on A and 1 on B, the player on B pays 6 TL and would pay 0.5 on A.
5. **Ratios.** *PoA = 6/2 = 3* (worst equilibrium), *PoS = 2/2 = 1* (best equilibrium).

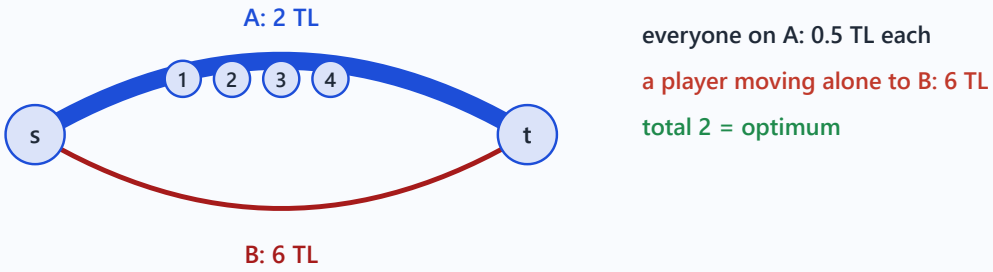


Figure 9.13: The good equilibrium: all four players on link A (2 TL), 0.5 TL each. A player moving alone to B would pay 6 TL. Total 2 = optimum.

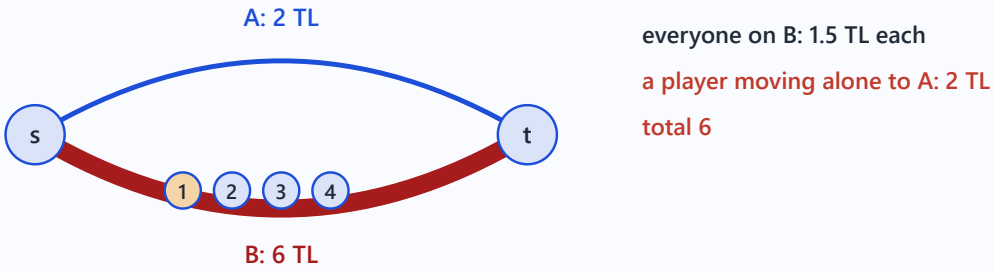


Figure 9.14: The bad equilibrium: all four players on link B (6 TL), 1.5 TL each. A player moving alone to A would pay 2 TL; nobody moves. Total 6.

Figure 9.13 and Figure 9.14 are two equilibria of the same game. At the bad equilibrium everyone would gain if all moved to A together (0.5 TL instead of 1.5). But a player who moves alone pays 2 TL and loses. The problem is a **coordination problem**: if all the players could move at the same time, the loss would disappear. The PoS is the measure of this optimistic scenario. If somebody tells the players "all of you use A" and nobody wants to deviate from the suggestion because it is an equilibrium, there is no loss. The PoA says how much can be lost if no suggestion is made.

How large can the PoA be? The same structure can be built with more extreme numbers. Let there be n players, a link of cost 1 and a link of cost n. If everyone is on the expensive link, each pays $n/n = 1$. A player who moved alone to the cheap link would also pay 1; the move is not profitable. This profile is an equilibrium and its total cost is n; the optimum is 1. So in fair cost sharing games the PoA can be as large as n. As the number of players grows, the worst equilibrium becomes unboundedly bad.

9.5.2 Step by step: the H_n example

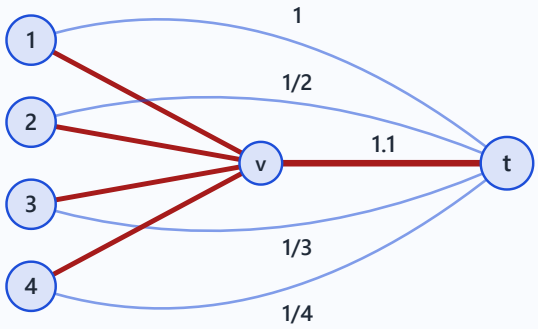
Since the PoA can be bad, the question turns to the PoS: is the best equilibrium always close to the optimum? The answer is "yes, but not exactly". In the example below even the best equilibrium is worse than the optimum by a certain ratio. The **harmonic number** appears in the example:

$$H_n = 1 + 1/2 + 1/3 + \ldots + 1/n$$

For example $H_1 = 1$, $H_2 = 3/2$, $H_3 = 11/6 \approx 1.83$, $H_4 = 25/12 \approx 2.08$. H_n grows very slowly and is approximately $\ln n$; $H_{100} \approx 5.19$.

Setup. There are 4 players and each goes to t by one of two routes: a route used only by himself or herself with cost $1/i$ (1 for player 1, 1/2 for player 2, 1/3 for player 3, 1/4 for player 4), or a shared route that everyone can use with cost $1 + \varepsilon = 1.1$. Here ε is a small extra cost (0.1). The cost of the shared route is split equally among its users.

Optimum. If everyone uses the shared route the total is 1.1. If everyone uses his or her own route the total is $1 + 1/2 + 1/3 + 1/4 = 25/12 \approx 2.083$. Mixed choices can be tried too, but once the shared route is used its cost is paid once, and those who add their own routes only increase the total. The optimum is the state in which everyone is on the shared route: total 1.1, 0.275 per person.



player	route	pays
1	shared	0.275
2	shared	0.275
3	shared	0.275
4	shared	0.275
total		1.1

Figure 9.15: The optimum in the H_n example: four players use the shared route (cost 1.1), each paying 0.275. The thin lines are the unused own routes (costs 1, 1/2, 1/3, 1/4).

Example 9.17

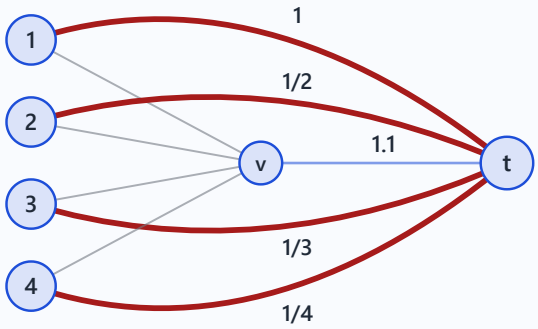
Start from the optimum and follow the best-response dynamics: at each step, if some player would gain by moving to his or her own route, that player moves.

1. **Player 4.** Pays 0.275 on the shared route; the own route costs $1/4 = 0.25 < 0.275$. Leaves. The remaining three now pay $1.1/3 \approx 0.367$.
2. **Player 3.** The own route costs $1/3 \approx 0.333 < 0.367$. Leaves. The remaining two pay $1.1/2 = 0.55$.
3. **Player 2.** The own route costs $1/2 = 0.5 < 0.55$. Leaves. Player 1 pays for the shared route alone: 1.1.
4. **Player 1.** The own route costs $1 < 1.1$. Leaves.

5. **The final state.** Everyone is on his or her own route. Nobody wants to go back to the shared route: a player who went back alone would pay 1.1, and 1.1 is more expensive than every player's own route. The total is *1 + 1/2 + 1/3 + 1/4 = 25/12 ≈ 2.083*.

The unique equilibrium. If $k \geq 1$ people are on the shared route, the highest-numbered of them has a number of at least k; its own route costs $1/i \leq 1/k < 1.1/k$, so it leaves. So no profile in which the shared route is used is an equilibrium, and the only equilibrium is "everyone on his or her own route". A script that scanned the 16 profiles confirmed this too.

Ratio. Since there is a single equilibrium, *PoS = PoA = (25/12) / 1.1 = 125/66 ≈ 1.894*.



player	route	pays
1	own	1
2	own	0.5
3	own	0.333
4	own	0.25
total		2.083

Figure 9.16: The unique equilibrium in the H_n example: everyone on his or her own route; the payments are 1, 0.5, 0.333 and 0.25, total *25/12 ≈ 2.083*.

The unraveling in the example is a chain reaction. The player whose cheapest alternative leaves the shared route first; this raises the share of the rest and pushes the next one to leave. As ε approaches zero the PoS of the same structure with n players approaches $H_n = 1 + 1/2 + \ldots + 1/n$: the optimum goes to 1 and the unique equilibrium to H_n. With four players this limit is *25/12 ≈ 2.08*; with ε = 0.1 we found 1.894.

9.5.3 Rosenthal's potential

The H_n example showed that the PoS can approach at least H_n. To show that this is an upper bound, that is, that the PoS exceeds H_n in no fair cost sharing game, an auxiliary function is needed.

Notation. c_e: the cost of link e; n_e: the number of players using e (each pays c_e/n_e). *H(k) = 1 + 1/2 + ... + 1/k* and *H(0) = 0*; $H_n = H(n)$. The total cost *C(s)* is the sum of the costs of the links used.

Definition 9.7

In a fair cost sharing game **Rosenthal's potential** is defined as

$$\Phi(s) = \Sigma_e c_e \cdot H(n_e)$$

The definition comes from an imaginary count. Think of the players as added to a link one by one: the 1st person writes c_e because he or she is alone, the 2nd writes $c_e/2$, ..., the k-th writes c_e/k. The sum of these numbers is $c_e \cdot (1 + 1/2 + \ldots + 1/k) = c_e \cdot H(k)$. Each person has written the share "at the moment of arrival"; the potential is the sum of these shares over all links.

The key property of the potential. When a player changes route, the change in his or her own cost equals exactly the change in the potential. The reason is this: when the player leaves a link e, the count of that link falls from n_e to $n_e - 1$ and the potential falls by exactly c_e/n_e (the last term of the sum), which is the share the player was paying there. When the player joins a new link e', the count of that link rises by one and $c_{e'}/(n_{e'} + 1)$ is added to the potential; this is the new share the player will pay there. So if the deviating player gains (the cost falls), the potential falls by the same amount. A game with such a function is called a **potential game**.

Theorem 9.5 (Potential and cost)

For every profile s, $C(s) \le \Phi(s) \le H_n \cdot C(s)$.

Proof. A link in use has at least one and at most n users; so $1 = H(1) \le H(n_e) \le H(n) = H_n$. Each term is multiplied by $c_e \ge 0$: $c_e \le c_e \cdot H(n_e) \le H_n \cdot c_e$. For links not in use, since $H(0) = 0$, the contribution to both the cost and the potential is zero. Summing over all links, the left side becomes $C(s)$, the middle $\Phi(s)$, and the right side $H_n \cdot C(s)$. □

Example 9.18

In the H_n example (four players, $\varepsilon = 0.1$):

1. **Everyone on the shared route.** The only link in use is the shared route and it has 4 users. $C = 1.1$, $\Phi = 1.1 \cdot H(4) = 1.1 \cdot 25/12 \approx 2.29$. $C \le \Phi$; moreover $\Phi = H_4 \cdot C$: equal to the upper bound, because all four players use the link.
2. **Everyone on his or her own route.** Each route has one user; $H(1) = 1$. $\Phi = C = 25/12 \approx 2.08$: equal to the lower bound.
3. **The effect of one move.** When player 4 leaves the optimum, his or her cost falls by $0.275 - 0.25 = 0.025$. The potential: the shared route $1.1 \cdot H(3) = 1.1 \cdot 11/6 \approx 2.017$, player 4's route $1/4 \cdot H(1) = 0.25$; total ≈ 2.267. The old potential was ≈ 2.292; a fall of ≈ 0.025 ✓.

9.5.4 PoS ≤ H_n: walking from the optimum to an equilibrium

Theorem 9.6 (Anshelevich et al., 2004)

In every network design game with fair cost sharing and n players, $PoS \le H_n$; in directed networks this bound is tight.

Proof. **1. Start from the optimum.** Run best-response dynamics from the profile s^{OPT}: at each step a player moves to a route that is cheaper for him or her.

2. Φ falls with every move. In a potential game the cost change of the deviating player equals the change of Φ; if the player gains, Φ falls. Since Φ falls strictly at each step, the same profile cannot be visited twice. Since the number of profiles is finite, the dynamics stop at some profile s. Where it stops, nobody can find a profitable move; s is a Nash equilibrium and $\Phi(s) \le \Phi(s^{OPT})$.

3. Build the chain.

$$C(s) \le \Phi(s) \le \Phi(s^{OPT}) \le H_n \cdot C(s^{OPT})$$

The first inequality is the left side of Theorem 9.5 (the potential is not smaller than the cost). The second is Step 2. The third is again the right side of Theorem 9.5 (the potential is at most H_n times the cost). So at least one equilibrium has cost at most H_n times the optimum: $PoS \le H_n$. Tightness comes from the H_n example. □

In the H_n example the numbers of the chain are: $C(s) \approx 2.08$, $\Phi(s) \approx 2.08$, $\Phi(OPT) \approx 2.29$, $H_4 \cdot C(OPT) = 25/12 \cdot 1.1 \approx 2.29$. That is, $2.08 \le 2.08 \le 2.29 \le 2.29$. The first and the third inequalities hold with equality; this is why the example comes so close to the bound.

The strength of the proof is that it does not need to know which equilibrium will be reached. The information “the potential falls” is enough. The same method does not work for the PoA: the worst equilibrium may lie where the dynamics from the optimum cannot reach, and its potential may be larger than that of the optimum. Indeed in these games the PoA can reach n while the PoS is at most $H_n \approx \ln n$. With 100 players this difference is 100 against about 5.19.

9.5.5 Common mistakes with the Price of Stability

Common mistakes

- **Assuming the optimum is always an equilibrium:** In the A/B example the optimum was an equilibrium; in the H_n example it is not. PoS = 1 only if the optimum is an equilibrium.
- **Confusing fair sharing with congestion:** Here crowding lowers the cost per person. In a congestion game crowding raises the delay. In the deviation computation the denominator must be $n_e + 1$ (including the newcomer).

- **Confusing the potential with the cost:** Φ is a computational tool, not a cost anyone pays. The total cost counts c_e for each link; the potential counts $c_e \cdot H(n_e)$.
- **Carrying the PoS bound over to the PoA:** H_n is a guarantee only for the best equilibrium. The worst equilibrium can be n times the optimum.

9.6 Reducing the PoA

Measuring the PoA is the first step; the second step is to reduce it. If the loss comes from an externality, the solution is to bring the externality into the player's own calculation. The most direct tool for this is the congestion toll; there are other tools too.

9.6.1 A congestion toll shifts the equilibrium

Let us take up the two-route network of 9.2.6 again: $\ell_1 = 5 + x_1/100$, $\ell_2 = 20 + x_2/200$, 3,000 cars in total. Without a toll the equilibrium is $x_1 = 2{,}000$ with a total of 75,000 car-min; the optimum is $x_1 = 1{,}500$ with a total of 71,250 car-min.

Let a **congestion toll** τ be put on Route 1. To keep the calculation simple, the toll is counted as an extra cost for the driver measured in minutes: a toll of $\tau = 2.5$ is the same, in the driver's eyes, as an extra delay of 2.5 minutes. The driver now sees $\ell_1 + \tau$ on Route 1 and at equilibrium $\ell_1 + \tau = \ell_2$:

1. $5 + \tau + x_1/100 = 20 + (3{,}000 - x_1)/200$.
2. Multiply both sides by 200: $1{,}000 + 200\tau + 2x_1 = 7{,}000 - x_1$.
3. $3x_1 = 6{,}000 - 200\tau \Rightarrow x_1 = (6{,}000 - 200\tau)/3$.

τ	x_1	x_2	Total delay (car-min)
0	2,000	1,000	75,000
2.5	≈ 1,833	≈ 1,167	≈ 72,917
5	≈ 1,667	≈ 1,333	≈ 71,667
7.5	1,500	1,500	71,250 (optimum)
10	≈ 1,333	≈ 1,667	≈ 71,667

Table 9.13: The equilibrium and the total delay as the toll τ (min) on Route 1 grows. The toll is not added to the total delay.

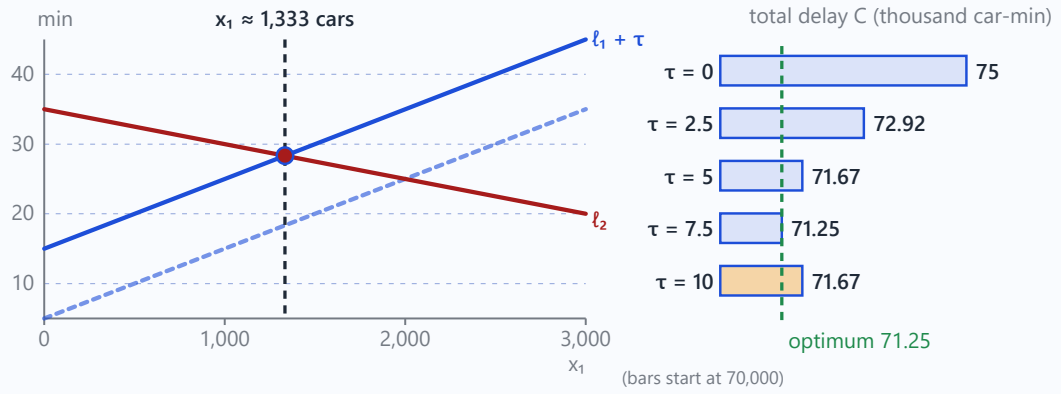


Figure 9.17: The effect of a toll in the two-route network. On the left the cost $\ell_1 + \tau$ of Route 1 as the driver sees it (blue) crosses ℓ_2 (red); as τ grows the crossing moves left. On the right the total delay for each τ; the smallest value is at τ = 7.5 (71,250), and at τ = 10 the total rises again.

Figure 9.17 shows the effect of the toll in two ways. In the graph on the left the curve of Route 1 shifts up by τ and the crossing of the two curves (the equilibrium) moves left: fewer drivers choose Route 1. The bars on the right show the total delay. As the toll grows some drivers move to the wide Route 2 and the total falls.

At τ = 7.5 the optimum becomes an equilibrium. $x_1 = (6{,}000 - 1{,}500)/3 = 1{,}500$, exactly the distribution of the optimum. The number 7.5 is not random. A marginal-cost toll puts on each route the externality on that route: on Route 1, $x_1 \cdot \ell_1' = 1{,}500 \cdot 1/100 = 15$ min; on Route 2, $x_2 \cdot \ell_2' = 1{,}500 \cdot 1/200 = 7.5$ min. If these two tolls are imposed, the driver sees $20 + 15 = 35$ on Route 1 and $27.5 + 7.5 = 35$ on Route 2; they are equal, so the optimum is an equilibrium. But the driver's decision is affected only by the difference between the two routes: $15 - 7.5 = 7.5$. That is why a toll worth 7.5 min on Route 1 alone gives the same result.

A toll is a transfer. The toll is collected from the drivers and goes elsewhere (to the municipality, to road building); it does not lengthen anyone's time. That is why it is not added to the total delay in the table. From the drivers' viewpoint, though, the situation is different: at τ = 7.5 the 1,500 drivers on Route 1 pay a total toll of $1{,}500 \cdot 7.5 = 11{,}250$ min. The drivers' "delay + toll" total is $71{,}250 + 11{,}250 = 82{,}500$, which is larger than the 75,000 of the untolled equilibrium. If the revenue collected is not redistributed to the drivers, the drivers as a whole lose; the gain passes to society (to wherever the revenue is used).

Too high a toll overshoots. At τ = 10, x_1 falls to about 1,333 and Route 2 fills more than it should; the total delay rises again to about 71,667. The right toll is the one equal to the externality; it is not "the more the better".

9.6.2 A marginal-cost toll fixes Pigou's network

In Pigou's network the aim is to make the optimum ($x = 1/2$) an equilibrium. Without a toll everyone is on the upper route and the total delay is 1.

Definition 9.8

A **marginal-cost toll** puts on each edge a toll equal to the externality of that edge: $\tau_e = f_e \cdot \ell_e'(f_e)$. The cost the driver sees is $\ell_e(f_e) + \tau_e$; this is the marginal cost of the edge.

Example 9.19

1. **Compute the tolls.** On the upper route $\ell(x) = x$, $\ell' = 1$; the toll is $\tau = x \cdot 1 = x$. On the lower route $\ell = 1$ is constant, $\ell' = 0$; there is no toll.
2. **What the driver sees.** $x + x = 2x$ on top, 1 on the bottom.
3. **The new equilibrium.** The costs they see are equalized: $2x = 1 \Rightarrow x = 1/2$. Selfish drivers choose the optimal distribution on their own. The total delay is $3/4$.
4. **The toll collected.** The 1/2 of the traffic on top each pay $\tau = 1/2$: $1/2 \cdot 1/2 = 1/4$.
5. **The drivers' bill.** Delay + toll $= 3/4 + 1/4 = 1$. The drivers pay the same total as at the untolled equilibrium (1); if the revenue is not redistributed the drivers do not gain, and the gain passes to society.

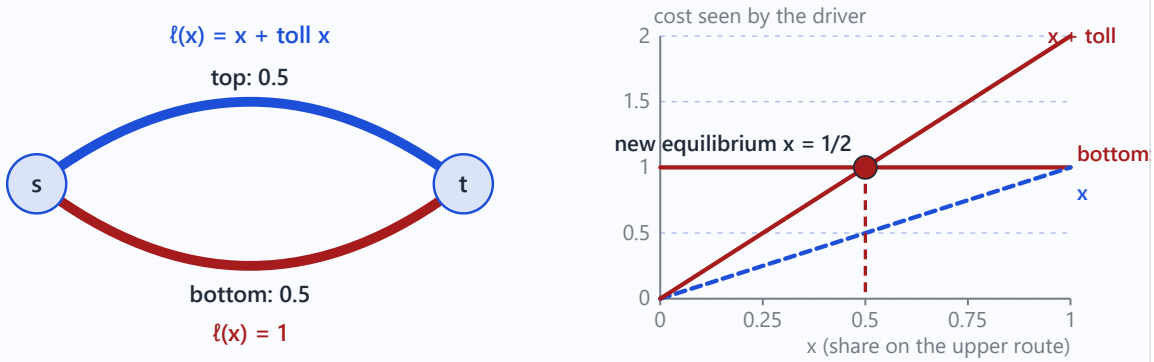


Figure 9.18: Pigou's network with a marginal-cost toll. On the upper route the cost the driver sees is $x + toll\ x = 2x$ (red sloping line), on the lower route 1 (red horizontal line). They cross at $x = 1/2$: the new equilibrium is the same as the optimum.

A fixed toll is enough too. The toll does not have to vary with traffic. If a fixed $\tau = 1/2$ is put on the upper route, the driver sees $x + 1/2$ on top; the equilibrium is $x + 1/2 = 1 \Rightarrow x = 1/2$. The right value of the fixed toll is the value of the marginal-cost toll at the optimum. This result generalizes:

Theorem 9.7 (Marginal-cost pricing)

In a network with nonatomic traffic, if the latencies are increasing and differentiable and the total cost $x \cdot \ell_e(x)$ of each edge is also convex (bowl-shaped) (linear and x^p latencies are like this), then when each edge is charged a toll equal to its externality at the optimum ($\tau_e = f_e^{OPT} \cdot \ell_e'(f_e^{OPT})$), the system optimum is a Wardrop equilibrium.

Proof. At the optimum all the paths carrying traffic have equal marginal costs and the marginal cost of unused paths is not smaller (9.2.5). Once the tolls are fixed at their values at the optimum, the cost a driver sees on an edge at the optimum is $\ell_e(f_e^{OPT}) + f_e^{OPT} \cdot \ell_e'(f_e^{OPT})$; this is exactly the marginal cost of the edge. Since the cost a path shows is the sum over its edges, the path costs the drivers see are the same as the marginal path costs at the optimum. So all the paths used at the optimum are equal and cheapest in the drivers' eyes: the Wardrop condition holds. □

The name of the toll also comes from Pigou: a tax that prices an externality is called a **Pigouvian tax**. In practice the difficulty is not in computing the toll but in measuring the externality: one must know the latency functions and the traffic.

9.6.3 Other ways to reduce the PoA

A toll is not the only tool. The main tools used in the literature and in practice are these:

Congestion pricing
: Brings the external cost that users impose on others into the decision. London (2003), Stockholm (a trial in 2006, permanent since 2007) and Singapore (since 1975) charge for entry to the city center.

Centralized (Stackelberg) routing
: A center routes part of the traffic and the rest chooses selfishly. The center moves first, and the selfish users respond by looking at its choice; this “leader first, then followers” structure takes its name from Stackelberg games. As the controlled share grows the ratio approaches 1 (Roughgarden, 2004).

Capacity planning
: Roughgarden and Tardos also showed this: the selfish equilibrium of r units of traffic is no more expensive than the best routing of $2r$ units of traffic. The result holds for every increasing latency, nonlinear ones included. In other words, making every link twice as fast makes up for the lack of coordination; adding capacity can do the job of a toll.

Route recommendations and information
: The recommendations of navigation apps can steer the equilibrium toward a lower total cost; but compliance is voluntary. If the recommendation is an equilibrium (nobody wants to deviate), compliance comes by itself.

Mechanism design
: Designing the rules so that the good outcome is the only equilibrium. For example, in the A/B link game, forbidding the rental of B or giving a small discount to whoever joins A removes the

bad equilibrium.

Counting the cost

The information, implementation and fairness costs of an intervention must be weighed together with the efficiency gain obtained. A toll system built to remove a 5% loss is pointless if it itself costs more than 5%.

Example 9.20

The capacity result can be tested in Pigou's network. The equilibrium of 1 unit of traffic is 1. For the optimum of 2 units of traffic the amount on the upper route x ($0 \le x \le 2$) is chosen: $C(x) = x \cdot x + (2 - x) \cdot 1$, $C'(x) = 2x - 1 = 0 \Rightarrow x = 1/2$, $C(1/2) = 1/4 + 3/2 = 7/4$. $1 \le 7/4$ ✓: the selfish equilibrium of one unit is cheaper than the best routing of two units.

9.6.4 Real life and discussion

Toll lanes in traffic networks, inter-domain route selection on the Internet, time-of-use tariffs in the power grid: the question is the same in all three. How bad is the selfish equilibrium, and which intervention brings it closer to the optimum? On the Internet every network operator routes traffic according to its own cost; this is a large-scale example of selfish routing. In the power grid consumers use electricity at the cheapest hour; a time-of-use tariff reflects the grid's externality at peak hours in the price.

The concepts of this chapter are consolidated by a few discussion questions:

- **Why do individual interests not always reach the social optimum? What externality is there in the Pigou example?** In Pigou's network a driver entering the upper route slows everyone there at the rate $\ell' = 1$; this extra delay ($x \cdot \ell' = x$) is not in his or her calculation. The optimum itself is also not stable: those on the bottom want to switch to the top.
- **Which incentives can be applied to reduce the PoA? Which is fairer?** A toll is efficient but can be regressive: the same toll weighs more heavily on a low-income driver. A bonus (a payment to whoever uses the lower route) can achieve the same distribution but requires a budget. Centralized routing “forces” some people onto the long route. The fairness debate comes down to the question of whose pocket the efficiency gain goes into.
- **Where can Braess's paradox be seen in everyday life? When does closing a road make sense?** It can occur in any network where a new link sends everyone to the same bottleneck. A frequently cited example is that after the demolition of the highway over the Cheonggyecheon in Seoul, which began in 2003, city traffic did not worsen as much as expected. Closing a road makes sense if it brings the equilibrium closer to the optimum; to tell, the equilibrium and the optimum must be computed together.

9.6.5 Common mistakes with tolls

Common mistakes

- **Adding the toll to the total delay:** A toll is a transfer; it does not change society's total delay. The drivers' "delay + toll" total is a separate number.
- **Thinking "the more the better":** $\tau = 10$ is worse than $\tau = 7.5$. The right toll equals the externality.
- **Forgetting the difference of the marginal tolls:** In the two-route network the tolls of the two routes are 15 and 7.5; the toll to be put on a single route is their difference (7.5).
- **Taking the revenue collected to be the toll:** In Pigou's network the toll per person is $\tau = x = 1/2$; 1/4 is the total revenue collected.
- **Putting a toll on a route with constant latency:** On the lower route $\ell' = 0$; there is no externality there and the marginal-cost toll is zero.

9.7 Worked problems

The problems in this section use all the methods of the chapter together: the optimum by derivative in Pigou-type networks, deviation checks in load balancing, the equilibrium in the Braess network, the H_n example, smoothness and code tracing. Each solution is given with its intermediate steps.

9.7.1 Sample problems

Example 9.21

Problem. Let the latency of the upper route in Pigou's network be x^2 (the lower route 1). What are the equilibrium cost, the optimum and the PoA?

Solution. Since $x^2 \le 1$, the upper route is never slower than the lower; the equilibrium is the state in which everyone is on top and its cost is 1. For the optimum, $C(x) = x \cdot x^2 + (1 - x) = x^3 - x + 1$, $C'(x) = 3x^2 - 1 = 0 \Rightarrow x^{OPT} = 1/\sqrt{3} \approx 0.577$. $C(OPT) = 1 - 2/(3\sqrt{3}) \approx 0.615$. $PoA \approx 1/0.615 \approx 1.626$; it is the same as the $p = 2$ row of Table 9.4.

Example 9.22

Problem. The jobs are {3, 3, 2, 2, 2}, 2 identical machines. What is the optimum makespan? Is there a Nash equilibrium with makespan 7?

Solution. The total is 12, so the optimum is at least 6; {3, 3} | {2, 2, 2} achieves this: OPT = 6. In the assignment {3, 2, 2} | {3, 2} the loads are 7 and 5. If a job of size 2 on M1 moves to M2 it waits $5 + 2 = 7$; not smaller than the present 7, so no gain. If the job of size 3 on M1 moves it waits 8. If the jobs on M2 move to M1 they wait 9 or 10. Nobody gains: an equilibrium, with ratio $7/6$.

Example 9.23

Problem. What is the PoA in the normalized Braess network ($s\to A$: x, $A\to t$: 1, $s\to B$: 1, $B\to t$: x, $A\to B$: 0; 1 unit of traffic)?

Solution. Equilibrium: everyone on the zigzag, $1 + 0 + 1 = 2$; a driver who switches to an old route also waits 2 and does not gain. Optimum: zigzag closed, traffic split half and half; each driver $1/2 + 1 = 3/2$. $PoA = 2 / (3/2) = 4/3$.

Example 9.24

Problem. If the H_n example is built for $n = 3$ (ε small), what does the PoS approach?

Solution. The own routes cost 1, 1/2, 1/3; the shared route $1 + \varepsilon$. With the same unraveling chain the only equilibrium is the state in which everyone is on his or her own route: $H_3 = 1 + 1/2 + 1/3 = 11/6 \approx 1.83$. Since the optimum is $1 + \varepsilon$, as $\varepsilon \to 0$, $PoS \to 11/6$.

Example 9.25

Problem. In Pigou's network with upper route $2x$ and lower route 1, what is the PoA and what fixed toll makes the optimum an equilibrium?

Solution. Equilibrium: $2x = 1 \Rightarrow x = 1/2$; the latency is 1 on both routes, $C = 1$. Optimum: $C(x) = 2x^2 + 1 - x$, $C' = 4x - 1 = 0 \Rightarrow x = 1/4$, $C = 1/8 + 3/4 = 7/8$. $PoA = 8/7$. Toll: at the optimum the driver on top waits $2 \cdot 1/4 = 1/2$; $1/2 + \tau = 1 \Rightarrow \tau = 1/2$. The marginal-cost toll is the same: $x \cdot \ell' = 1/4 \cdot 2 = 1/2$.

9.7.2 Traces and new topics

Example 9.26

Problem. What does `pigou_poa(3)` return?

Solution. $x = 4^{-1/3} \approx 0.63$; $x^4 \approx 0.1575$, $1 - x \approx 0.37$; $C(OPT) \approx 0.528$. The function returns $1/0.528 \approx 1.896$; the value lies between $p = 2$ (1.626) and $p = 4$ (2.151).

Example 9.27

Problem. Four jobs of sizes 4, 3, 3, 2 all start on M1. At each step the first job with a profitable deviation moves to the least loaded machine. At which makespan do the dynamics stop? What is the optimum?

Solution. The loads are (12, 0). The job of size 4 moves to M2 ($0 + 4 < 12$): (8, 4). Then a job of size 3 moves ($4 + 3 = 7 < 8$): (5, 7). Now if the job of size 4 on M2 moved to M1 it would wait 9, if a job of size 3 moved it would wait 8 (both larger than 7); if those on M1 moved to M2 they would wait 10 or 9. The dynamics stop at makespan 7. The optimum is {4, 2} | {3, 3} = 6 | 6; the ratio is $7/6$.

Example 9.28

Problem. In the game of Example 9.16 let link A cost 3 TL and link B cost 8 TL (4 players). Which are the equilibria, and what are the PoA and the PoS?

Solution. All on A: 0.75 each; a player who moves to B pays 8, an equilibrium. All on B: 2 each; a player who moves to A pays 3, an equilibrium. In split profiles those on B gain by moving to A (for example with 1 on A and 3 on B, the player on B pays 8/3 and would pay 1.5 on A). $PoA = 8/3$, $PoS = 1$.

Example 9.29

Problem. What does `poa_pos([3, 2, 2, 1], 2)` return?

Solution. The total is 8; {3, 1} | {2, 2} = 4 | 4 is the optimum. {3, 2} | {2, 1} (loads 5 and 3) is also an equilibrium: if the job of size 3 moves it waits 6, if a job of size 2 moves it waits $3 + 2 = 5$ (equal, no gain); if those on M2 move they wait 7 or 6. The maximum loads of the equilibria are 4 and 5: the function returns `(1.25, 1.0)`, that is, $PoA = 5/4$, $PoS = 1$.

Two exercises on smoothness, the bound of a $(1, \tfrac{1}{2})$-smooth game (2) and the check of the key inequality for $(y, z) = (2, 3)$ ($8 \le 29/3$), are solved in Example 9.8 and Example 9.9.

9.8 Summary

- The **Price of Anarchy** divides the cost of the worst Nash equilibrium by the cost of the social optimum: $PoA = \max_{s \in NE} C(s) / C(OPT)$. The **Price of Stability** writes the same ratio for the best equilibrium. Always $1 \le PoS \le PoA$; if there is a single equilibrium the two are equal. In payoff problems the ratio is inverted.

- The PoA is a **pessimistic guarantee** given without knowing which equilibrium will be reached. It is the game-theoretic counterpart of the approximation ratio: in the approximation ratio computational power is what is missing, in the PoA it is coordination.
- **In Pigou's network** (upper route x, lower route 1) the equilibrium sends everyone to the top and costs 1; the optimum splits the traffic in half and costs 3/4: *PoA = 4/3*.
- At equilibrium the **latencies** of the routes are equalized; at the optimum their **marginal costs** ($\ell + f \cdot \ell'$) are. The term $f \cdot \ell'$ in between is the externality that a driver imposes on the others.
- When the latency gets steeper, like x^p, the PoA grows without bound (≈ 4.73 at $p = 16$). **With linear latency** $PoA \le 4/3$ in every network and for every demand (Roughgarden–Tardos); the proof consists of the equilibrium condition, the edge inequality and summation; the bound is tight in Pigou's network.
- In **(λ, μ)-smooth** games $PoA \le \lambda/(1 - \mu)$; nonatomic linear traffic gives 4/3 with *(1, ¼)*, atomic linear congestion games give 5/2 with *(5/3, 1/3)*. The bound carries over to mixed equilibria (robust PoA).
- **In the Braess network** (4,000 cars) the equilibrium is 320,000 and the optimum of the same network is 258,750 car-min: $PoA \approx 1.237$. 80/65 is the Braess loss, not the PoA.
- In the **load balancing** game a pure equilibrium always exists. In the {2, 2, 1, 1} example *PoA = 4/3*, *PoS = 1*; on m identical machines the tight value is $2 - 2/(m + 1)$. On machines with different speeds and at mixed equilibria the ratio grows slowly with m. The value of the PoA depends on the social cost measured.
- **In fair cost sharing** the PoA can reach n, but thanks to Rosenthal's potential $\Phi = \Sigma\, c_e\, H(n_e)$ we have $PoS \le H_n \approx \ln n$; the H_n example shows that the bound is tight.
- A **marginal-cost toll** $\tau_e = f_e \cdot \ell_e'(f_e)$ makes the optimum an equilibrium (in Pigou's network $\tau = x$ or a fixed 1/2; in the two-route network 7.5 min on Route 1). A toll is a transfer; too high a toll creates loss again. Stackelberg routing, added capacity, information and mechanism design are other tools.

Example	Equilibrium	Optimum	Ratio
Two routes, 3,000 cars	75,000	71,250	PoA ≈ 1.053
Pigou (x and 1)	1	3/4	PoA = 4/3
Linear latency, any network	—	—	PoA ≤ 4/3 (tight)
Pigou, $\ell = x^p$	1	→ 0	PoA → ∞
Atomic congestion, linear	—	—	PoA ≤ 5/2 (tight)
Braess, 4,000 cars	320,000	258,750	PoA ≈ 1.237
Load balancing {2, 2, 1, 1}	4 (worst)	3	PoA = 4/3, PoS = 1
Fair sharing, H_n example	25/12	1.1	PoS ≈ 1.894 (≤ H_n)
Pigou + toll τ = x	3/4	3/4	1

Table 9.14: The examples of the chapter at a glance.

9.9 Key terms

Price of Anarchy (PoA) Price of Stability (PoS) coordination ratio worst-case equilibrium social optimum social cost efficiency loss approximation ratio tight bound Pigou network latency function nonatomic atomic Wardrop equilibrium system optimum selfish routing marginal cost externality congestion game smoothness, (λ, μ)-smooth game robust Price of Anarchy Braess's paradox load balancing makespan related machines best-response dynamics fair cost sharing network design harmonic number H_n potential game Rosenthal potential congestion toll marginal-cost toll congestion pricing Pigouvian tax Stackelberg routing capacity augmentation

9.10 Test Yourself

1. 1 unit of traffic is split between two parallel routes: the latency of Route 1 is $\ell_1(x) = x$ and that of Route 2 is $\ell_2(x) = x + 1/2$. What are the Wardrop equilibrium, the social optimum and the PoA?

   Answer

   Equilibrium: equal latencies, $x_1 = (1 - x_1) + 1/2 \Rightarrow x_1 = 3/4$, $x_2 = 1/4$; everyone waits 3/4, $C = 3/4$. Optimum: equal marginal costs, $2x_1 = 2x_2 + 1/2$ and $x_1 + x_2 = 1 \Rightarrow x_1 = 5/8$, $x_2 = 3/8$. $C = (5/8)^2 + (3/8)(3/8 + 1/2) = 25/64 + 21/64 = 23/32$. $PoA = (3/4)/(23/32) = 24/23 \approx 1.043$, far below 4/3.

2. In Pigou's network (upper route x, lower route 1; here x is the amount of traffic on the upper route) let the total traffic be $r = 1/2$ and $r = 2$. What is the PoA in each case?

   Answer

   $r = 1/2$: the latency of the upper route is at most 1/2, everyone is on top; $C = 1/2 \cdot 1/2 = 1/4$. The optimum is $C(x) = x^2 + (1/2 - x)$; since the derivative $2x - 1$ is always ≤ 0 on the interval, the smallest value is at $x = 1/2$: again 1/4. $PoA = 1$; with little traffic the externality is unimportant. $r = 2$: the equilibrium is 1 unit on the upper route (latency 1) and 1 unit on the lower; $C = 1 + 1 = 2$. The optimum is $C(x) = x^2 + 2 - x$, $x = 1/2$, $C = 7/4$. $PoA = 8/7 \approx 1.143$.

3. For the edge inequality $\ell(f) \cdot f^{OPT} \le \ell(f^{OPT}) \cdot f^{OPT} + \tfrac{1}{4} \cdot \ell(f) \cdot f$ take $\ell(x) = x$ and $f = 1$. What are the

two sides for $f^{OPT} = 1/4$? At which value of f^{OPT} does equality hold?

Answer

Left $1 \cdot 1/4 = 1/4$; right $1/16 + 1/4 = 5/16$. $1/4 \leq 5/16$ ✓; the difference is $(1/4 - 1/2)^2 = 1/16$. The difference is always $(f^{OPT} - f/2)^2$; equality holds only at $f^{OPT} = f/2 = 1/2$. Pigou's network uses exactly this point.

4. A game is known to be $(2, 1/3)$-smooth. What bound results for the PoA? Would this information be useful if the same game were $(1, 1)$-smooth?

Answer

$PoA \leq 2/(1 - 1/3) = 2/(2/3) = 3$. $(1, 1)$ is not a pair allowed by the definition ($\mu < 1$ is required). In Step 4 of the proof the inequality $(1 - \mu) \cdot C(s) \leq \lambda \cdot C(OPT)$ becomes $0 \leq C(OPT)$, which is always true and gives no bound.

5. The jobs are {5, 4, 3, 2}, 2 identical machines, the social cost is the maximum load. What is the optimum? Is {5, 4} | {3, 2} a Nash equilibrium? What are the PoA and the PoS?

Answer

The total is 14, so the maximum load is at least 7; {5, 2} | {4, 3} = 7 | 7: OPT = 7. In {5, 4} | {3, 2} the loads are 9 and 5. If the job of size 5 moves it waits $5 + 5 = 10$, if the job of size 4 moves it waits $5 + 4 = 9$ (equal, no gain); if those on M2 move they wait 12 or 11. Equilibrium ✓, maximum load 9. Scanning the 16 assignments shows that the maximum loads of the equilibria are 7, 8 and 9: $PoA = 9/7 \approx 1.286 \leq 4/3$, $PoS = 1$ (7 | 7 is also an equilibrium).

6. 3 players rent a connection from s to t: A costs 1 TL, B costs 2 TL; those who use a link split its cost equally. Which are the pure equilibria? What are the PoA and the PoS?

Answer

All on A: 1/3 each; a player who moves to B pays 2, an equilibrium. All on B: 2/3 each; a player who moves to A pays 1 ($1 > 2/3$), an equilibrium. 1 on A, 2 on B: the one on B pays 1 and would pay 1/2 on A; leaves. 2 on A, 1 on B: the one on B pays 2 and would pay 1/3 on A; leaves. The equilibria are “all on A” (total 1 = OPT) and “all on B” (total 2): $PoA = 2$, $PoS = 1$.

7. Build the H_n example with two players: own routes 1 and 1/2, shared route 1.2 (ε = 0.2). What are

the optimum, the unique equilibrium and the PoS? How are $\Phi(OPT)$ and $\Phi(equilibrium)$ related?

Answer

Optimum: both on the shared route, total 1.2 (the other options are 1.5, 1.7 and 2.2). From the optimum: player 2 pays 0.6 on the shared route, its own route costs 0.5 < 0.6, and it leaves; player 1 pays the whole cost of the shared route (1.2) alone, its own route costs 1 < 1.2, and it leaves. Equilibrium: everyone on his or her own route, $C = 1.5$. $PoS = 1.5/1.2 = 1.25 \le H_2 = 1.5$. Potential: $\Phi(OPT) = 1.2 \cdot H(2) = 1.8$, $\Phi(equilibrium) = 1 + 1/2 = 1.5$; along the dynamics the potential has fallen from 1.8 to 1.5.

8. 2,000 cars split between two routes: $\ell_1 = 10 + x_1/100$, $\ell_2 = 15 + x_2/100$ (min). What are the total delays at the equilibrium and at the optimum? What fixed toll on Route 1 makes the optimum an equilibrium?

Answer

Equilibrium: $10 + x_1/100 = 15 + (2{,}000 - x_1)/100 \Rightarrow x_1 = 1{,}250$, everyone 22.5 min; total 45,000 car-min. Optimum: $10 + x_1/50 = 15 + x_2/50 \Rightarrow x_1 - x_2 = 250$, $x_1 = 1{,}125$, $x_2 = 875$; latencies 21.25 and 23.75; total $1{,}125 \cdot 21.25 + 875 \cdot 23.75 = 44{,}687.5$. PoA ≈ 1.007. Toll: $21.25 + \tau = 23.75 \Rightarrow \tau = 2.5$ min; this equals the difference of the marginal tolls ($11.25 - 8.75 = 2.5$).

9.11 Further reading

- T. Roughgarden, *Twenty Lectures on Algorithmic Game Theory* (2016), Lectures 11–15: covers all the topics of this chapter, from the Pigou and Braess examples to smoothness and the H_n bound in network design, in short and clear lectures.
- N. Nisan, T. Roughgarden, É. Tardos and V. Vazirani (eds.), *Algorithmic Game Theory* (2007), Chapters 17–20: comprehensive reference chapters on load balancing, selfish routing, network formation and the Price of Stability.
- D. Easley and J. Kleinberg, *Networks, Crowds, and Markets* (2010), Chapter 8: explains equilibrium in traffic and Braess's paradox with examples for readers with little mathematics.
- A. R. Karlin and Y. Peres, *Game Theory, Alive* (2017), Chapter 8: presents the Price of Anarchy together with potential games and congestion games, with proofs at the student level.

9.12 References

1. E. Anshelevich vd., "The Price of Stability for Network Design with Fair Cost Allocation", *IEEE FOCS*, 2004.
2. B. Awerbuch, Y. Azar, A. Epstein, "The Price of Routing Unsplittable Flow", *ACM STOC*, 2005.
3. D. Braess, "Über ein Paradoxon aus der Verkehrsplanung", *Unternehmensforschung* 12, 258–268, 1968.

4. G. Christodoulou, E. Koutsoupias, “The Price of Anarchy of Finite Congestion Games”, *ACM STOC*, 2005.
5. A. Czumaj, B. Vöcking, “Tight Bounds for Worst-Case Equilibria”, *ACM-SIAM SODA*, 2002.
6. D. Easley, J. Kleinberg, *Networks, Crowds, and Markets*, Cambridge, 2010 (Chapter 8: traffic and Braess).
7. G. Finn, E. Horowitz, “A Linear Time Approximation Algorithm for Multiprocessor Scheduling”, *BIT* 19, 1979.
8. R. L. Graham, “Bounds for Certain Multiprocessing Anomalies”, *Bell System Technical Journal* 45(9), 1966.
9. A. R. Karlin, Y. Peres, *Game Theory, Alive*, AMS, 2017 (Chapter 8: The Price of Anarchy).
10. E. Koutsoupias, M. Mavronicolas, P. Spirakis, “Approximate Equilibria and Ball Fusion”, *Theory of Computing Systems* 36(6), 2003.
11. E. Koutsoupias, C. Papadimitriou, “Worst-case Equilibria”, *STACS*, 1999.
12. N. Nisan, T. Roughgarden, É. Tardos, V. Vazirani (eds.), *Algorithmic Game Theory*, Cambridge University Press, 2007 (Chapters 17–20).
13. C. Papadimitriou, “Algorithms, Games, and the Internet”, *ACM STOC*, 2001.
14. A. C. Pigou, *The Economics of Welfare*, Macmillan, 1920.
15. T. Roughgarden, “The Price of Anarchy Is Independent of the Network Topology”, *Journal of Computer and System Sciences*, 2003.
16. T. Roughgarden, “Stackelberg Scheduling Strategies”, *SIAM Journal on Computing* 33(2), 2004.
17. T. Roughgarden, “Intrinsic Robustness of the Price of Anarchy”, *ACM STOC*, 2009.
18. T. Roughgarden, *Twenty Lectures on Algorithmic Game Theory*, Cambridge University Press, 2016 (Lectures 11–15).
19. T. Roughgarden, É. Tardos, “How Bad Is Selfish Routing?”, *Journal of the ACM*, 2002.
20. J. G. Wardrop, “Some Theoretical Aspects of Road Traffic Research”, *Proceedings of the Institution of Civil Engineers* 1(3), 325–362, 1952.

PART IV

Cooperation and Society

Chapters 10–12

Multi-Agent Systems

In a warehouse, hundreds of robots pick orders; in a search-and-rescue team, drones divide up regions; in a smart grid, households buy and sell energy. In each of these systems there are several decision makers, each with its own information and its own goal. We call such decision makers **agents** and the system as a whole a **multi-agent system**. Because agents are affected by one another's decisions, the interaction among them is a game. Starting from the notion of an agent, these notes cover cooperation and coordination problems, task allocation with the contract net, coalitional games (the Shapley value, the core, power indices), and free riding in the public goods game. For every topic a numerical example is worked through with all intermediate steps.

LEARNING OUTCOMES

Agents and MAS
Explain the basic properties of an agent and the problems of cooperation, competition, and coordination in multi-agent systems.

Task allocation
Run the contract net protocol (announcement, bid, award) on a small example and compare its result with the optimum.

Coalitional games
Write down a characteristic function; compute the Shapley value from marginal contributions over all orderings; test whether an allocation is in the core; find the power index in weighted voting.

Public goods and free riding
Show numerically why contributing nothing is a dominant strategy in the public goods game, and recognize its counterparts in multi-agent systems.

10.1 Agents and multi-agent systems

In the earlier chapters the players were mostly people: buyers bidding in an auction, voters casting ballots, users shopping in a marketplace. In this chapter a large share of the players are software: robots, drones, autonomous vehicles, trading programs. We call such software **agents**. Agents do not

think like humans, but like humans they pursue their own goals and are affected by the decisions of other agents. For this reason the tools of game theory apply to them directly.

10.1.1 A warm-up game: the Shapley coalition game

Three players, A, B, and C, will work together to produce some value. Each player can produce a certain value alone. When two players work together, a **synergy bonus** arises on top of the sum of their stand-alone values. The rule of the game is this: the value of a coalition (a group working together) is the sum of its members' stand-alone values plus the synergy bonus of every pair inside the coalition. Finally, the value of the coalition of all three players, *v(N)*, is divided among the players.

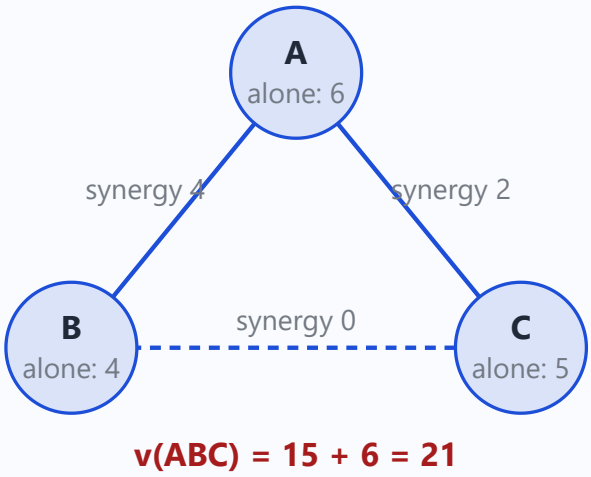


Figure 10.1: One sample round of the Shapley Coalition Game. The nodes show the stand-alone values (A 6, B 4, C 5) and the edges the pairwise synergy bonuses (A–B 4, A–C 2, B–C 0). The value of the grand coalition is *v(ABC) = 15 + 6 = 21*.

Figure 10.1 shows a sample round. The stand-alone values add up to *6 + 4 + 5 = 15* and the synergies to *4 + 2 + 0 = 6*. Working all together, the players produce *v(ABC) = 15 + 6 = 21* units. The game poses two questions:

- If the 21 units are split equally, everyone gets 7. Is A treated unfairly by this split?
- Should the most valuable player alone always receive the largest share?

An equal split looks fair at first glance. But A is both the most valuable player alone (6) and a partner in two synergies (4 and 2). B contributes to only one synergy (4), and C to only one (2). For a split to count as "fair" we need a rule that measures each player's contribution. That rule is the Shapley value, defined in 10.4.5. The answer to the game is computed in 10.4.8: A's Shapley share is 9, so the equal split gives A 2 units less than A's contribution warrants. The answer to the second question is also "no" in general. What determines the split is not the stand-alone value but the average of the value a player adds to all possible groups.

This game combines the two big themes of the chapter. The first is that agents working together produce value that none could produce alone (cooperation). The second is how that value is divided among the agents (sharing). If the division is done badly, some agents want to leave the group and the cooperation falls apart. The rest of the chapter builds on these two themes.

10.1.2 What is a multi-agent system?

Definition 10.1

A **multi-agent system** (MAS) is a system made up of several autonomous agents that perceive, decide, and act in the same environment. Each agent observes the environment with its own sensors, makes its own decisions, and changes the environment through its own actions. The agents may share a common goal, or each agent may compete with the others for its own interest.

Three parts of the definition matter. The first is "several": the problem of a single agent is a decision problem, while the problem of several agents is a game. The second is "autonomous": agents make their decisions themselves rather than taking orders from a center. The third is "the same environment": an agent's action changes the environment, and this change affects the other agents' perceptions and payoffs.

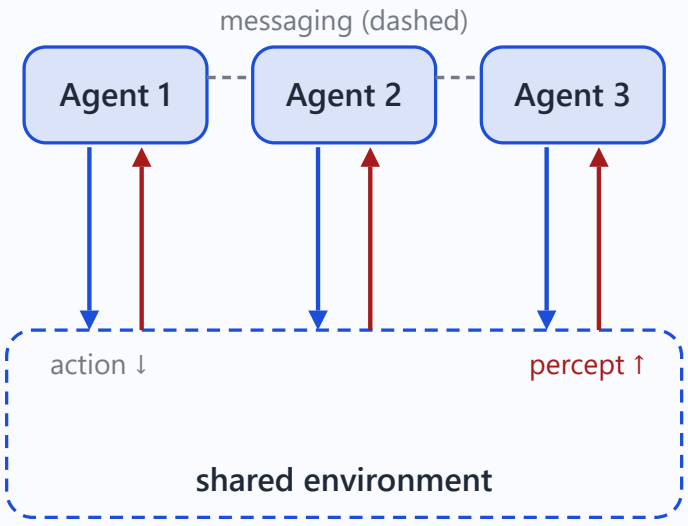


Figure 10.2: A multi-agent system. Three agents apply actions to a shared environment (blue arrows, downward) and receive percepts from it (red arrows, upward). The dashed lines between the agents show message passing.

Figure 10.2 summarizes this structure. Each agent has a two-way link to the environment: perception and action. Agents can also exchange messages with one another. No agent in the figure sees the whole environment. **Every agent's information is local.** A warehouse robot sees only the shelves around it. A drone scans only the region it flies over. An autonomous vehicle senses only the vehicles near it. For this reason two design problems always arise in multi-agent systems:

- **Coordination:** the agents' decisions must be compatible with one another. Two robots must not go to the same shelf, and two drones must not scan the same region.
- **Communication:** agents fill in their missing information with messages. But messaging has a cost (time, energy, bandwidth), and messages can be delayed or lost.

Typical examples of multi-agent systems are robot teams, autonomous vehicle fleets, distributed sensor networks, and traffic or economic simulations. In all of them the number of agents is large, information is scattered, and having a single center compute everything is either impossible or too expensive.

The connection to game theory. If agents are affected by one another's decisions, the interaction among them is a game. The four elements of a game map directly onto a multi-agent system: the players are the agents, the strategies are the agents' action choices, the payoffs are the utility functions that measure the agents' goals, and the information structure is who observes what. The person who designs the agents takes on two roles. They program the agents' insides (the decision rule), but they also determine the rules the agents face (who gets what, who pays what). The second role is that of the **mechanism designer**, who sets the rules. Mechanism design means building the rules of the game so that players, pursuing their own interests, choose the desired outcome.

10.1.3 The agent concept and its four properties

In everyday language an "agent" is someone who acts on another's behalf. In artificial intelligence, an agent is a piece of software or a machine that acts on its own in an environment. Wooldridge and Jennings (1995) gather the most basic properties a system needs in order to count as an agent into four items. This is called the **weak notion of agency**; "weak" indicates that the definition does not include human-like aspects of the agent such as consciousness or intention, and only describes its observable behavior.

Definition 10.2

An **agent** is a computer system with the following four properties:

- **Autonomy**: it makes its own decisions without human intervention and manages its own internal state.
- **Reactivity**: it perceives changes in the environment and responds to them in a timely way.
- **Pro-activeness**: it does not merely respond; it takes the initiative and plans in order to reach its goal.
- **Social ability**: it communicates, negotiates, and cooperates with other agents.

The four properties are easiest to see on a single agent. Consider a search-and-rescue drone. Table 10.1 matches each property with one behavior of this drone.

Property	Meaning	Drone example
Autonomy	Makes its own decisions	Chooses its own route
Reactivity	Responds to change in time	Brakes when it sees an obstacle
Pro-activeness	Takes initiative toward a goal	Heads to the station before its battery runs out
Social ability	Interacts with other agents	Bids for a task

Table 10.1: The four properties of the weak notion of agency and their counterparts in a drone.

The difference between reactivity and pro-activeness is subtle but important. Reactive behavior is triggered by an event: an obstacle appears and the drone brakes. Without the event there would be no behavior. Pro-active behavior is triggered by a goal: the drone computes that its charge will not last for the rest of the route and heads for the station before any warning has appeared. A pro-active agent anticipates the future; a reactive agent answers the present. A good agent balances the two. A purely reactive agent misses long-term goals. A purely pro-active agent may stick to its plan so firmly that it cannot respond in time to an unexpected obstacle.

Social ability is the property that separates a multi-agent system from a single-agent one. Agents without social ability affect one another only through the environment, even when they share it. Socially able agents exchange messages, make bids, and strike agreements. Task allocation (10.3) and coalition formation (10.4) rest on this ability.

The agent in game theory. Game theory models one aspect of an agent in detail: its decision. In game theory an agent is a **rational player**. Its preferences are expressed by a utility function, and the agent tries to maximize its expected utility. Here "expected" refers to the case where the outcome is uncertain: the utility of each possible outcome is weighted by its probability and summed. For example, a drone prefers a route that yields a utility of 10 or 4 with equal probability over a route that yields a sure 6, because the expected utility of the first route is $0.5 \cdot 10 + 0.5 \cdot 4 = 7$.

10.1.4 How does an agent decide? Architectures

The four properties describe *what* an agent does. *How* the agent reaches its decision is determined by its **agent architecture**. The architecture is the internal path from perception to action. Table 10.2 compares five common architectures.

Architecture	Decision mechanism	Strength	Example
Reactive	State → action rules; no internal model	Fast, robust	Robot vacuum, swarm behavior
Deliberative	World model + planning / search	Long-term goals	Exploration robot, logistics planner
BDI	Belief, desire, intention	Human-like reasoning	Air traffic management, virtual assistant
Learning	Learns a policy from experience	Adapts to unknown environments	Game-playing agents
Hybrid	Reactive lower layer + deliberative upper layer	Speed and foresight together	Autonomous vehicle

Table 10.2: Agent architectures.

The **reactive architecture** is the simplest. Inside the agent is a rule table: "if the way ahead is clear, move forward; if you hit a wall, turn randomly." The agent keeps no model of the environment and computes nothing about the future. For this reason it is very fast and robust to sensor errors. Household robot vacuums come close to this architecture. Swarm behavior is also the product of reactive rules: Reynolds (1987) showed that realistic flocks of birds emerge when each bird follows just three local rules (avoid collisions with neighbors, fly in the same direction as neighbors, move toward the center of the neighbors).

The **deliberative architecture** keeps a model of the environment and searches over this model to produce a sequence of actions (a plan). An exploration robot stores the map in memory and computes the shortest path to its goal. Planning makes it possible to reach long-term goals, but it is slow. If the environment changes before the plan is finished, the plan may become invalid.

The **BDI architecture** organizes planning around three notions that resemble human reasoning. *Beliefs* are what the agent holds to be true about the environment ("runway 3 is free"). *Desires* are the states the agent would like to reach ("land the aircraft", "save fuel"). Desires can conflict with one another. *Intentions* are the goals the agent has chosen from among its desires and has committed to achieving ("land the aircraft on runway 3"). Intention keeps the agent from changing its mind at every moment: the agent stays committed to its intention until new information signals an important change.

The **learning architecture** does not have its rules written in advance; it works out from experience which action gives good results in which situation. The learned rule is a **policy**: a function that assigns to each state an action (or a probability distribution over actions). This architecture adapts to unknown environments. Game-playing agents mostly work this way.

The **hybrid architecture** combines layers. The lower layer of an autonomous vehicle is reactive: if a pedestrian suddenly steps in front of it, it brakes without waiting for the planner. Its upper layer is a planner: it computes the route and lane changes ahead of time. In this way speed and foresight are obtained together.

One distinction must be kept in mind: **the architecture describes the inside of the agent, whereas game theory studies the interaction between agents.** A game-theoretic model does not care whether the agent contains reactive rules or BDI logic. It looks only at which strategies the agent can choose and what it earns from each outcome. In the rest of the chapter, agents are treated at this abstract level, that is, through their strategies and payoffs.

10.1.5 Real-world applications

Multi-agent systems are used today in many fields. Each application connects to a game-theoretic question later in the chapter.

Autonomous vehicles
: Coordination is needed for the crossing order at intersections, lane changes, and convoy driving. If two vehicles try to enter the same lane at the same moment, there is a risk of collision. This is a coordination game (10.2.5).

Drone teams
: In search and rescue the regions must be shared out: which drone will scan which region? This is a task allocation problem (10.3).

Warehouse robots
: Hundreds of robots pick orders. Routes must be collision-free, and jobs must be distributed evenly among the robots.

Online marketplaces
: Buyer and seller agents post price offers and negotiate automatically. This is the territory of auctions and mechanism design.

Smart grid
: Energy is traded among households, storage units, and producers, and demand is managed. If everyone draws energy at the peak hour, the grid is strained; this is an incentive problem (10.2.6).

Simulation
: In traffic, epidemic, and economic models, individuals are represented as agents. The simple rules of the individuals can produce unexpected patterns at the system level (10.2.7).

All of these applications ask the same question: **when agents with their own goals come together, how do we obtain a good outcome at the system level?** What counts as a "good outcome" depends on the context. A short total delivery time in a warehouse, a low total energy cost in a grid, and a fair division of the gain in a coalition are all good outcomes. Game theory separates two questions. The analysis question is where the agents end up when they act in their own interest (the equilibrium). The design question is how the rules should be set up so that the point they reach is a good one (the mechanism).

10.1.6 Common mistakes about agents

Common mistake

Confusing pro-activeness with reactivity. The behavior "the drone returns to the station before its battery runs out" is not reactivity but pro-activeness. There is no external event triggering the drone; it acts on its own goal to prevent a problem that will arise later. An example of reactivity is "braking when it sees an obstacle".

Taking a convention to be a property of an agent. A rule such as "everyone drives on the right" (a convention) is not a property of a single agent but a coordination rule shared among agents.

Thinking an agent is only a physical robot. A trading program, a bidding bot, or a virtual assistant is also an agent. Being an agent does not require a body; the four properties are enough.

Mixing up architecture and game theory. Writing an agent with a BDI architecture does not prevent it from being modeled as a rational player. The architecture describes the inside of the agent, and the game model describes what lies between agents.

10.2 Cooperation and coordination

How much do the agents' goals overlap? This question determines which problem has to be solved in a multi-agent system. If the goals are exactly the same, the problem is purely one of coordination: everyone wants the same thing, but the decisions have to be made compatible. If the goals are exactly opposed, the problem is one of strategy: each agent tries to predict its opponent's move and to make itself unpredictable. Most real systems lie between these two extremes.

10.2.1 Cooperative, competitive, and mixed systems

Multi-agent systems fall into three types according to the relationship among the agents' goals.

Cooperative MAS
: The agents have a common goal. They share information, tasks, and resources. The central problem is **coordination**: who does what, and when? Example: a search-and-rescue drone team. Every drone wants to find the missing person; it does not matter which drone finds them.

Competitive MAS
: The agents have different objectives. Each agent tries to increase its own payoff while taking the others' behavior into account. The central problem is **strategy**. Example: bidder agents in an auction. An item that one bidder wins is lost by another.

Mixed-motive MAS

Some goals are shared, while some resources are contested. Example: households sharing the same electricity grid. All of them want the grid not to collapse (a common goal), but they compete for the limited energy in the cheap hours (competition). Most real systems are of this type.

The question of which structure is more efficient has no single answer. Table 10.3 summarizes the conditions under which each extreme is efficient.

Structure	Condition under which it is efficient
Cooperation	Goals overlap, communication is cheap, and there is work no single agent can do
Competition	There is private information that can be revealed through prices or bids (market, auction)

Table 10.3: When are cooperation and competition more efficient?

Cooperation is indispensable for work that no single agent can do, such as drones carrying a heavy load together. The advantage of competition concerns information. If only the agent itself knows its own cost, the center cannot see this information directly. An auction or a tender pushes the agents to reveal it through their bids: the lowest-cost agent submits the lowest bid and gets the job. Competition thus becomes a tool for gathering information. This is exactly what mechanism design does: it sets up rules that steer competition toward a good outcome. Which structure to choose depends on the purpose of the system and on the structure of its resources.

10.2.2 The spectrum of interests: three small games

The three types of system can be represented by three small two-player games. Each game is given by a **payoff matrix**: the rows are Player 1's options, the columns are Player 2's options, and each cell lists (Player 1's payoff, Player 2's payoff). A cell is a **Nash equilibrium** when both players are at a **best response** in that cell. That is, with the other side's choice held fixed, neither player can raise their payoff by changing their own choice alone.

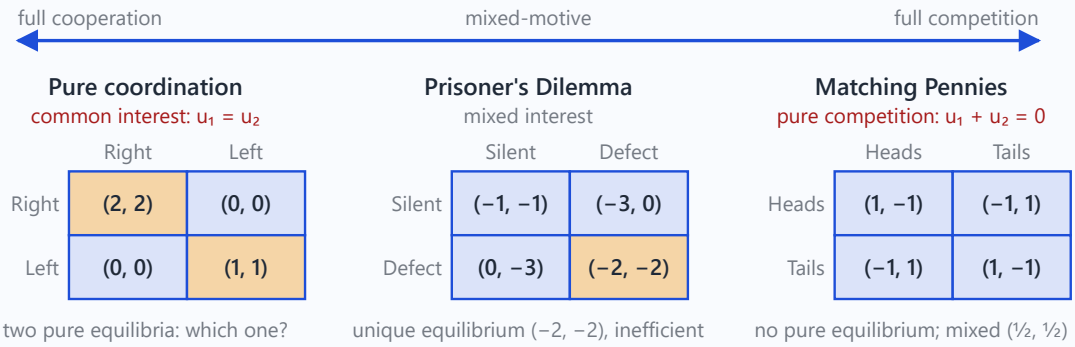


Figure 10.3: The spectrum of interests. On the left, pure coordination (common interest, $u_1 = u_2$); in the middle, the Prisoner's Dilemma (mixed interest); on the right, Matching Pennies (pure competition, $u_1 + u_2 = 0$). Orange cells are pure Nash equilibria.

Figure 10.3 places the three games on a spectrum running from full cooperation to full competition. Each game stands for one type of problem.

Left: pure coordination. The two players' payoffs are equal in every cell ($u_1 = u_2$). Interests are entirely shared. If both choose Right, both get 2; if both choose Left, both get 1; if they choose differently, both get 0. There is no conflict, but there is a problem: if the players do not know which side the other will choose, they may make incompatible choices. The solution is communication or a common rule (a convention, e.g., "everyone keeps right").

Middle: the Prisoner's Dilemma. Interests partly conflict. If both players stay silent, both get −1. If one defects and the other stays silent, the defector gets 0 and the silent one gets −3. If both defect, both get −2. Here the individual incentive runs against the common good, and a solution requires incentive design (10.2.3).

Right: Matching Pennies. Interests are completely opposed: one player's gain is the other's loss ($u_1 + u_2 = 0$, a zero-sum game). If the two players choose the same face, Player 1 wins; if they choose different faces, Player 2 wins. What matters here is being unpredictable. The way to achieve it is a **mixed strategy**: choosing each option with a certain probability, for instance by flipping a coin.

Example 10.1

The pure Nash equilibria of the three games are found by checking, in each cell, whether both players are at a best response.

1. **Pure coordination, (Right, Right) = (2, 2).** With Player 2 at Right, if Player 1 switches to Left, Player 1 gets 0; since 2 > 0, Player 1 does not switch. The game is symmetric, so Player 2 does not switch either. This cell is an equilibrium.
2. **Pure coordination, (Left, Left) = (1, 1).** A player who switches to Right gets 0; since 1 > 0, nobody switches. This cell is also an equilibrium, but it is worse for both than (Right, Right).
3. **Pure coordination, (Right, Left) and (Left, Right).** Both get 0; a player who changes sides matches the other and gets 1 or 2. These cells are not equilibria. Conclusion: there are two pure equilibria, and the problem is which one will be played.
4. **Prisoner's Dilemma.** If the other side stays Silent, Defect gives 0 and Silent gives −1: Defect is better. If the other side Defects, Defect gives −2 and Silent gives −3: Defect is again better. Defect is better whatever the other side does. The only equilibrium is (Defect, Defect) = (−2, −2).
5. **Matching Pennies.** At (Heads, Heads), Player 2 gets −1 and gets 1 by switching to Tails. At (Heads, Tails), Player 1 gets −1 and gets 1 by switching to Tails. In each of the four cells, the losing player wants to change their choice. There is no pure equilibrium.
6. **The mixed equilibrium in Matching Pennies.** Let Player 2 play Heads with probability q. Player 1's expected payoff from Heads is $q \cdot 1 + (1 - q) \cdot (-1) = 2q - 1$, and from Tails it is $q \cdot (-1) + (1 - q) \cdot 1 = 1 - 2q$. For Player 1 to be indifferent between the two options we

need $2q - 1 = 1 - 2q$, that is, $q = 1/2$. By symmetry Player 1 also plays $p = 1/2$. The equilibrium is (½, ½), and each player's expected payoff is 0.

Example 10.1 exposes three different problems at three points of the spectrum. In the coordination game there are several equilibria, and the problem is selection. In the Prisoner's Dilemma the equilibrium is unique but inefficient. In Matching Pennies there is no pure equilibrium and randomness is required. A person designing a multi-agent system cannot choose the right tool without understanding where on this spectrum the system lies. A coordination problem is solved with a common rule. An incentive problem, in contrast, is not solved by a common rule alone; it needs an arrangement that changes the payoffs.

10.2.3 Individual rationality and collective harm

The Prisoner's Dilemma carries the most important warning for multi-agent systems. In this game Defect is a **dominant strategy**: whatever the other side does, Defect yields the higher payoff. So both rational players choose Defect, and the equilibrium is (−2, −2). Yet (Silent, Silent) would have given both (−1, −1). Each player does the best for themselves, but the result is worse for both. **Individual rationality produces a collectively bad outcome.**

This situation is expressed through the notion of **Pareto optimality**. An outcome is Pareto optimal if there is no other outcome that improves at least one person's situation without making anyone worse off. (Defect, Defect) is not Pareto optimal: moving to (Silent, Silent) improves both players. The lesson of the Prisoner's Dilemma is that a Nash equilibrium need not be Pareto optimal.

In multi-agent systems this structure has many counterparts.

Bandwidth
: When every agent raises its own data rate, its own transfer seems to get faster, but the network congests and becomes slower for everyone.

Shared resource
: A shared processor, energy supply, or data pool is overused. Each agent puts its own job first, and the common resource is exhausted.

Information sharing
: Sharing information improves the common model, but withholding it gives the agent a short-term advantage.

The many-player version of this structure is the **public goods game**, which is examined numerically in 10.5. The exhaustion of a common resource through overuse is called the **tragedy of the commons**. In both the logic is the same: the individual's gain and the group's gain move in opposite directions.

10.2.4 Decision making: centralized or distributed?

Agents plan on the basis of what they know about the environment. Who makes the decision: a single center, or each agent itself? This is the first big decision in designing a multi-agent system. Table 10.4 compares the two options.

	Centralized control	Distributed (full autonomy)
Advantage	All information at one point: the optimum can be computed; easy coordination, low risk of conflict	Flexible, scalable, fast response to local events; the system keeps working when one part fails
Disadvantage	Single point of failure and bottleneck; computation explodes as the number of agents grows; agents may not report private information honestly	Decisions made on local information may be inconsistent; selfish agents may settle at an inefficient equilibrium (the Price of Anarchy)
Example	Traffic-light control center; central server in a sensor network	Drone teams, autonomous vehicles, robotic exploration

Table 10.4: Centralized versus distributed control.

The greatest strength of centralized control is that a center holding all the information can compute the optimum. But it has three weaknesses. The first is the **single point of failure**: if the center breaks down, the whole system stops. The second is **scaling**: as the number of agents grows, so do the messages arriving at the center and the problem the center has to solve. For example, trying every way of assigning n agents to n tasks means evaluating n! assignments one by one (10.3.5); this number reaches 3,628,800 at n = 10. The third is **honesty**: the center depends on the agents to report private information such as costs. A selfish agent may report a high cost in order to get an easier job.

Distributed control removes these weaknesses: if one agent breaks down the others keep working, adding a new agent is easy, and every agent reacts to local events immediately. But it has weaknesses of its own. Decisions made with local information may be inconsistent (10.2.5). Selfish agents may also reach an inefficient equilibrium, as in the Prisoner's Dilemma. The system-level cost of selfish behavior is measured by the ratio of the cost of the worst equilibrium to the cost of the optimum. This ratio is called the **Price of Anarchy**.

Most real multi-agent systems are **hybrid**: agents decide autonomously, but a center sets the rules, prices, or tenders. The center does not decide what each agent will do; it only sets the rules of the game. The contract net described in 10.3.2 is an example: a manager puts tasks out to tender, but it does not know the costs and leaves the choice of who does the job to the bids.

10.2.5 Distributed coordination and conventions

In distributed coordination, agents decide without a center, using local information and messaging. A protocol that achieves this must handle three things together. **Communication delay**: by the time a message arrives, the situation may have changed. **Fault tolerance**: if a message is lost or an agent does not respond, the system must not lock up. **Scalability**: as the number of agents grows, the number of messages must not get out of control.

The most typical problem of a distributed system is the lack of common knowledge. Suppose two robots meet in a narrow corridor. Each will step to one side to let the other pass. If both step to their own right, their paths separate and they get by. If one steps to its right and the other to its left, they head for the same side and again end up nose to nose. This situation is written as a game in Table 10.5.

R1 \ R2	Right	Left
Right	**(1, 1)**	(−1, −1)
Left	(−1, −1)	**(1, 1)**

Table 10.5: The corridor game. The two pure Nash equilibria are (Right, Right) and (Left, Left).

At (Right, Right), a robot that switches to Left on its own drops to −1, so nobody deviates. The same holds for (Left, Left). In the other two cells both get −1 and the robot that changes sides rises to 1; these cells are not equilibria. Both equilibria are equally good. The problem is not which one is better but **which one will be played**.

Definition 10.3

A **convention** or **social law** is a common rule, known to all agents, that selects one of the equilibria of a coordination game in advance. The rule "everyone keeps right" selects the (Right, Right) equilibrium in the corridor game.

The strength of a convention is that it is self-enforcing. If everyone keeps right, keeping right is each agent's best response; an agent that breaks the rule hurts only itself. This is why no police are needed to enforce a convention. The source of this strength is that a convention selects an equilibrium; a rule that selects a non-equilibrium outcome does not enforce itself.

Example 10.2

In the corridor game, the expected payoffs of three situations are compared: no convention (mixed strategy), a convention, and a common signal.

1. **No convention.** The robots do not know which side the other will choose. The only reasonable symmetric behavior is to choose each side with probability ½. If R2 chooses Right with probability q, R1's expected payoff from Right is $q - (1 - q) = 2q - 1$, and from Left it is $-q + (1 - q) = 1 - 2q$. Indifference gives $q = 1/2$. This is a mixed Nash equilibrium.
2. **Payoff at the mixed equilibrium.** The robots choose the same side with probability ½ (each gets 1) and different sides with probability ½ (each gets −1). The expected payoff is $\tfrac{1}{2} \cdot 1 + \tfrac{1}{2} \cdot (-1) = 0$. Half of the encounters end in a collision.

3. **With a convention.** Under the rule "everyone keeps right", both robots choose Right and each gets exactly 1. The payoff rises from 0 to 1.
4. **With a common signal.** Let a common signal, like a traffic light, say "both right" or "both left" with probability one half each. A robot that knows the other will go right when the signal says "right" has right as its best response; the same holds for "left". Nobody wants to ignore the signal, and each robot again gets 1.

The last step of Example 10.2 shows an important notion. A traffic light gives everyone a common signal and selects which of the two equilibria is played. An equilibrium of this kind, in which following the signal is the best response for every player, is called a **correlated equilibrium**. A correlated equilibrium arises when the players are given an outside "recommendation" and following it is in each player's own interest. The traffic light at an intersection is an everyday example.

10.2.6 Game theory and incentives

Game theory does two separate jobs in multi-agent systems.

Analysis: the Nash equilibrium
: Where do the agents' strategic decisions lead at equilibrium? Is the equilibrium efficient? This is predicting the outcome once the rules of the system are given.

Design: the mechanism
: How do rules, payments, and auctions steer agents toward honest reporting and cooperation? This is choosing the rules that will produce the desired outcome.

The power of design shows in two examples.

Energy sharing. In a smart grid, some households produce energy with solar panels. If the incentive is set up in a balanced way, for instance with a fair payment for surplus production fed into the grid, households feed the surplus into the grid and everyone gets enough energy. If the incentive is set up wrongly, for instance if the same price applies at every hour, everyone draws energy at the hour that suits them best, namely the evening peak, and the grid is strained.

Distributed network resources. Bandwidth and computing power are shared resources. Without usage fees or quotas, every agent uses the resource to the fullest for its own work, and overuse results. A fee or quota makes each agent feel the cost that its own use imposes on others.

The conclusion from the two examples is this. The **right incentive** increases cooperation and reduces conflict. The **wrong incentive** breeds selfish behavior, excessive risk-taking, and free riding. Incentive design means setting up rules under which selfish behavior is also good for the system, instead of expecting agents not to be selfish.

10.2.7 Communication, learning, and emergent behavior

In the basic models of game theory, the players know the game and play the equilibrium at once. In multi-agent systems three real-world circumstances complicate this simple picture.

Communication and information. Agents have different information. One robot knows that a shelf is empty; another does not. Who knows what, that is, the **information structure**, changes the equilibrium. Games in which players do not fully know one another's payoffs or types are called games of incomplete information. For agents to exchange information they need a common language and a messaging protocol. The contract net (10.3.2) is such a protocol.

Learning and adaptation. Instead of computing the equilibrium in advance, agents can update their strategies step by step from feedback. There are three common methods. In *best-response dynamics*, each agent plays a best response to the others' latest move. In *fictitious play*, each agent plays a best response to the average of its opponent's past moves. In *reinforcement learning*, the agent chooses actions that have given good results more often. But there is a problem: because the other agents are learning too, each agent's environment keeps changing. Such an environment is called **non-stationary**. In a non-stationary environment there is no guarantee that learning converges to an equilibrium.

Example 10.3

In the corridor game (Table 10.5), let the two robots start from the cell (Right, Left); both have payoff −1. Two different update rules are compared.

1. **Simultaneous best response, round 1.** R1 plays a best response to R2's last choice, Left: Left. At the same time R2 plays a best response to R1's last choice, Right: Right. The new cell is (Left, Right), and both again get −1.
2. **Simultaneous best response, round 2.** R1 chooses Right as the best response to Right, and R2 chooses Left as the best response to Left. The cell becomes (Right, Left) again.
3. **Result.** The two robots oscillate forever between (Right, Left) and (Left, Right). Each tries to adapt to the other, but because the other changes at the same moment, the adaptation never happens. The dynamics do not converge.
4. **Sequential best response.** This time let only R1 update. R1 chooses Left as the best response to R2's Left. The cell becomes (Left, Left). When R2's turn comes, R2 is already at a best response and does not change. The dynamics reach an equilibrium in a single step.

Example 10.3 makes the consequence of a non-stationary environment concrete. For each robot the environment is the behavior of the other robot; since the other robot is learning too, this environment changes in every round. Putting the updates in sequence (or adding a small random wait for the robots) breaks the cycle. The practice in network protocols of having colliding messages wait a random time and resend rests on the same idea.

Emergent behavior. Simple local rules can produce unexpected patterns at the system level. This is called **emergent behavior**. The most famous example is Schelling's (1971) segregation model. Individuals from two groups are placed on a grid. Each individual moves to an empty spot if the share of its own group among its neighbors falls below a small threshold (for instance below one third). No individual wants segregation; everyone is content to live in a mixed neighborhood. Even so, the moves cascade, and in the end sharply segregated neighborhoods form. A mild preference about neighbors produces strong segregation that nobody wanted.

This example carries a methodological lesson: **the behavior of the system cannot be read off by looking at the agents one by one.** This is why equilibrium analysis and simulation are used together in multi-agent systems. Equilibrium analysis gives the stationary points the system can reach; simulation shows how the system travels to these points, or whether it gets there at all. Traffic jams and price fluctuations are further patterns of this kind that arise from the interaction among agents.

10.2.8 Common mistakes in coordination

Common mistake

Looking for the one equilibrium of a game with several. The corridor game has two pure equilibria: (Right, Right) and (Left, Left). There is also a ½–½ mixed equilibrium. The answer to "how many pure equilibria are there?" is 2; the mixed equilibrium is not added to this count.

Taking a coordination problem for an incentive problem. In the corridor game there is no conflict of interest; both robots want to avoid a collision. No payment or penalty is needed here; a common rule is enough. In the Prisoner's Dilemma, by contrast, a common rule ("let us both stay silent") does not enforce itself, because the rule does not select an equilibrium.

Taking the commonly good outcome to be an equilibrium. (Silent, Silent) is better for both players than (Defect, Defect), but it is not an equilibrium: a player who switches to Defect while the other is silent gets 0 instead of −1.

Assuming centralized control is always superior. If the center knows all the information it can compute the optimum, but it has to get this information from the agents, and the agents may not report it honestly. The center is also a single point of failure.

10.3 Task allocation

The most common concrete problem a multi-agent system faces is distributing jobs among agents. Which drone will scan which region, which robot will pick which order, which server will do which computation? The contract net protocol, one of the oldest distributed methods, can be compared with the centralized optimum on a small example.

10.3.1 Task and resource allocation

Definition 10.4

Task allocation is the problem of distributing jobs among suitable agents, taking abilities and costs into account. The goal is usually to optimize some aggregate measure, for example to minimize the total time. When what is being distributed is not jobs but a resource such as processor time, bandwidth, or energy, the problem is called **resource allocation**.

On paper the problem looks simple: write down in a table how long each agent needs for each job, and look for the assignment with the smallest total. In reality there are three difficulties.

- **Private information.** Often only the agent itself knows the costs. How many minutes a robot needs to pick an order depends on the robot's position, charge, and load. The center cannot see this table.
- **Scale.** The number of agents is large. There may be hundreds of robots and thousands of orders.
- **Time.** Tasks arrive over time. Not all tasks are known up front; decisions have to be made as new orders come in.

A good method balances solution quality against communication and computation cost. Computing for hours or sending thousands of messages to find a perfect assignment can be more expensive than finding a slightly worse assignment instantly. Table 10.6 compares four common methods.

Method	How it works	Pros / cons
Centralized optimization	All costs are collected and the assignment problem is solved (Hungarian algorithm)	Optimal / needs all information
Contract net	A task is announced, agents bid, the best bid wins	Simple, distributed / can be greedy
Auction	Tasks or resources are sold by bidding; bundles in a combinatorial auction	Can be designed to be incentive compatible
Market-based	Prices are adjusted to supply and demand, agents react to prices	Scales / convergence can be slow

Table 10.6: Task allocation methods.

Centralized optimization collects all costs at a center and solves the **assignment problem**: assign n tasks to n agents so that each agent gets exactly one task and the total cost is minimal. The classical solution of this problem is the **Hungarian algorithm**. The Hungarian algorithm finds the optimum in about n^3 steps; that is, about a thousand steps suffice for n = 10. What the method lacks is not computation but information: the center must know the entire cost table.

The **contract net** does not gather information at the center; it puts tasks out to tender one at a time. **Auction** methods generalize this idea. In a **combinatorial auction**, agents bid not on single tasks but on bundles of tasks: "I can do G1 and G2 together in 5 minutes." Bundle bids capture

complementarities among tasks (such as picking two nearby orders together). In **market-based** methods each task or resource has a price. The price rises when demand is in excess and falls when it is low; agents decide which task to take by looking at the price. This method scales to very large systems, but it can take time for prices to settle at equilibrium.

10.3.2 The contract net protocol

The Contract Net Protocol was proposed by Smith (1980) for distributed problem solving. The protocol imitates the tendering process in the construction industry: a client announces the job, contractors submit bids, and the job is given to the best bid. FIPA (the Foundation for Intelligent Physical Agents), which sets agent standards, also published the protocol as a standard in 2002 (FIPA, 2002).

Definition 10.5

In the **contract net protocol**, the agent that has a task plays the role of **manager**, and the agents that can do the job play the role of **contractor**. Four steps are followed for each task:

1. **Task announcement**: the manager broadcasts the task and the bid deadline to the contractors.
2. **Bid**: each eligible contractor sends a bid stating its cost or capability.
3. **Award**: the manager compares the bids and awards the contract to the best one.
4. **Result report**: when the contractor finishes the job, it reports the result to the manager.

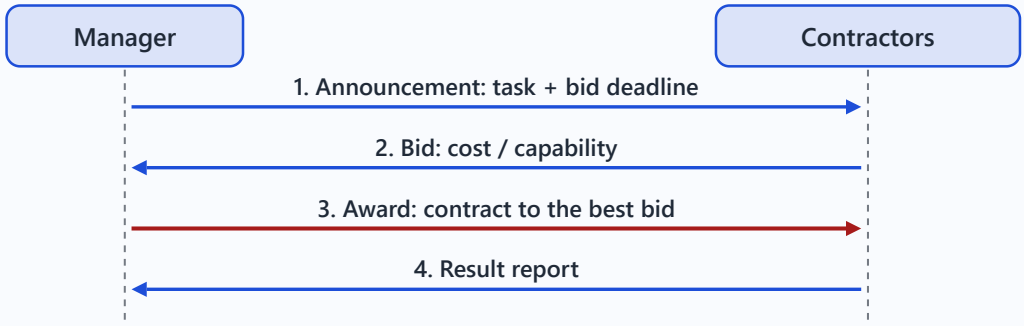


Figure 10.4: The message sequence of the contract net protocol. Four messages pass between the manager and the contractors: announcement, bid, award (red), and result report.

Figure 10.4 shows the order of the messages. The order cannot be changed: a task that has not been announced cannot be bid on, and an award cannot be made before the bids are collected. The roles are not fixed; they **can change from task to task**. A robot that has taken on a task as contractor can put a sub-part of that task out to tender to other robots and become the manager for that subtask. In this way a large job is broken into pieces by nested tenders.

The most important property of the protocol concerns information. **The manager does not know the costs; the bids reveal this private information.** The manager does not have to learn the whole cost table; for each task it sees only the bids for that task. Bidding is a **reverse auction**: in an ordinary auction the highest bid wins, whereas here the job goes to whoever reports the lowest cost. This structure raises the question of whether bidders will be honest; that question is taken up in 10.3.6.

10.3.3 The contract net step by step

Let us run the protocol on a small example. A manager will put three tasks (G1, G2, G3) out to tender in turn. There are three robots (R1, R2, R3). Each robot can take at most one task. Each robot's estimated time (in minutes) for each task is given in Figure 10.5. These times are the robots' costs, and the robots report them honestly as bids. An important detail: **only the robots know the table; the manager does not see it.** Each robot knows only its own column.



Figure 10.5: The initial state. On the left are the manager and the three robots; on the right is the cost table (minutes). Rows are tasks and columns are robots: for example R1 does G1 in 2 minutes, G2 in 3 minutes, and G3 in 5 minutes. The table is drawn faded because the manager cannot see it.

The critical point of the table is this: R1 is the cheapest robot for both G1 (2) and G2 (3). Since R1 can take only one task, which task it goes to will determine the total time.

Example 10.4

1. **Announcement of G1.** The manager broadcasts G1 to all idle robots: R1, R2, R3.
2. **Bids for G1.** Each robot reports its own cost for G1: R1 → 2, R2 → 3, R3 → 6.
3. **Award of G1.** The lowest bid is R1's (2 min). G1 goes to R1. R1 is now busy. Total time 2 min.
4. **Announcement of G2.** The manager broadcasts G2 to the idle robots: R2, R3. R1 does not receive the announcement because it is busy.
5. **Bids for G2.** R2 → 7, R3 → 8. Although R1 (3 min) is the cheapest robot for G2, it cannot bid.

6. **Award of G2.** The lowest bid is R2's (7 min). G2 goes to R2. Total time *2 + 7 = 9* min.
7. **Announcement of G3.** Only R3 remains idle.
8. **Bid for G3.** R3 → 9. There is only one bid.
9. **Award of G3.** G3 goes to R3; since only one bid is left, the choice is forced. Total time *9 + 9 = 18* min.

The contract net finds the assignment G1→R1, G2→R2, G3→R3, and the total time is 18 minutes.

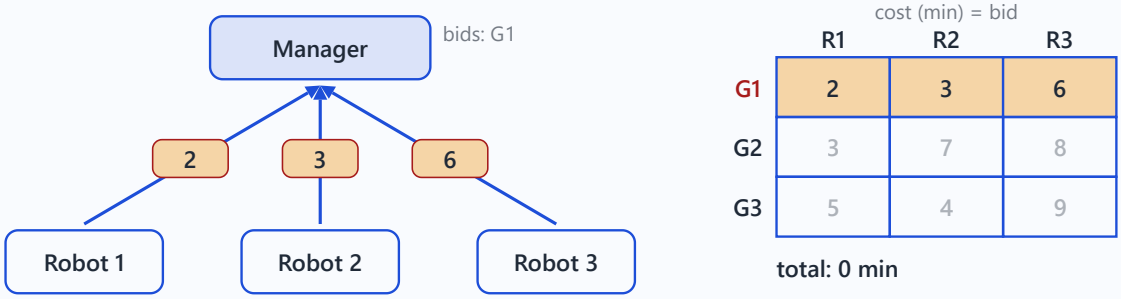


Figure 10.6: Bids for G1. All three robots bid (orange boxes 2, 3, 6); the G1 row of the table is highlighted.



Figure 10.7: The award of G1. The thick red arrow shows the contract going to R1, and the red frame in the table marks the winning bid. Total 2 min.

Figure 10.6 and Figure 10.7 show the bid and award steps of the first tender. The next two tenders proceed in the same way; the only difference is that after each award the number of idle robots drops by one. Table 10.7 summarizes the three tenders together.

Announcement	Idle robots	Bids	Award	Total
G1	R1, R2, R3	R1 2, R2 3, R3 6	R1 (2)	2
G2	R2, R3	R2 7, R3 8	R2 (7)	9

Announcement	Idle robots	Bids	Award	Total
G3	R3	R3 9	R3 (9)	18

Table 10.7: The three tenders of the contract net.



Figure 10.8: The result. Red frames show the contract net's assignment (total 18 min); dashed blue frames show the centralized optimum: G1→R3, G2→R1, G3→R2, total 13 min.

Figure 10.8 compares the result with the optimum. A center that tries all assignments gives G1 to R3 (6), G2 to R1 (3), and G3 to R2 (4), and obtains a total of *6 + 3 + 4 = 13* minutes. The contract net's 18 minutes is 5 minutes above the optimum. As a ratio, *18/13 ≈ 1.385*; that is, the contract net is about **38% more expensive**.

Where does the difference come from? At each step the contract net makes the best decision of the moment: it gives G1 to the cheapest robot, R1. This decision is right for G1, but it uses up R1 for G2. Although R1 is the cheapest robot for G2, G2 goes to R2 at 7 minutes. The optimum, by contrast, makes a somewhat expensive choice for G1 (R3, 6 min) but saves R1 for G2. Methods that make the choice that looks locally best at every step are called **greedy** methods. Greedy methods are fast, but because they do not take the future into account they can miss the overall optimum.

10.3.4 Code trace: contract_net()

The sequential tender logic of the contract net can be written as a Python function of a few lines. The function takes two inputs. `tasks` is the list of tasks in the order in which they are put out to tender: `["G1", "G2", "G3"]`. `cost` is a nested dictionary: `cost[r][g]` is robot r's cost for task g. In the example, `cost["R1"] = {"G1": 2, "G2": 3, "G3": 5}`.

```
def contract_net(tasks, cost):
  free = set(cost)
  assign, total = {}, 0
  for g in tasks:
    bids = {r: cost[r][g] for r in free}
    winner = min(bids, key=bids.get)
    assign[g] = winner
    total += bids[winner]
    free.remove(winner)
  return assign, total
```

The code works line by line as follows.

1. `free = set(cost)`: Building a set from a dictionary takes the dictionary's keys. `free` initially contains all the robots: {R1, R2, R3}.
2. `assign, total = {}, 0`: The assignments are kept in an empty dictionary, and the total time starts at 0.
3. `for g in tasks`: The tasks are put out to tender in order. Each iteration is one announcement.
4. `bids = {r: cost[r][g] for r in free}`: Only the idle robots bid. Each robot reports its own cost for task g. This line is the bid step of the protocol.
5. `winner = min(bids, key=bids.get)`: `min` picks, among the dictionary's keys (the robots), the one with the smallest `bids.get` value. This line is the award step.
6. `assign[g] = winner` and `total += bids[winner]`: The task is recorded for the winner, and the winning bid is added to the total.
7. `free.remove(winner)`: The winning robot becomes busy and does not take part in later tenders.

g	free (start of iteration)	bids	winner	assign	total
—	{R1, R2, R3}	—	—	{}	0
G1	{R1, R2, R3}	{R1: 2, R2: 3, R3: 6}	R1	{G1: R1}	2
G2	{R2, R3}	{R2: 7, R3: 8}	R2	{G1: R1, G2: R2}	9
G3	{R3}	{R3: 9}	R3	{G1: R1, G2: R2, G3: R3}	18

Table 10.8: Trace of `contract_net(["G1", "G2", "G3"], cost)`. Each row is the state at the end of one iteration of the loop.

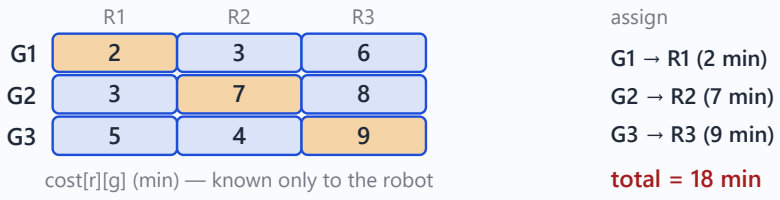


Figure 10.9: The final state of the `contract_net()` trace. Orange cells are the winning bids; on the right are the assignment list and the total of 18 min.

Table 10.8 and Figure 10.9 summarize the run of the code on the example. The most instructive part of the trace is the G2 iteration: in this iteration R1 is not in the set `free`. R1, which took G1 cheaply, was also the cheapest candidate for G2 (3), but it is now busy. At each step the function makes the locally best decision, but the result is not the overall optimum. The cost of the code is also low: for m tasks and n robots, at most n bids are compared in each iteration, so the total work is about $m \cdot n$ steps.

One more detail of the code matters: in case of a tie, the `min` function picks the first robot it meets in the dictionary. If two robots submit the same bid, the winner depends on the internal order of the collection. In a real protocol the tie-breaking rule (for example, the lower-numbered robot wins) should be stated explicitly. No tie occurs in this example.

10.3.5 The centralized optimum: optimum()

To judge the result of the contract net we need to know the optimum. A center that knows all the costs can try every assignment of robots to tasks. For three tasks and three robots, an assignment is an ordering of the robots: the first robot in the ordering takes G1, the second takes G2, and the third takes G3. The three robots have $3! = 3 \cdot 2 \cdot 1 = 6$ different orderings.

```
from itertools import permutations
def optimum(tasks, cost):
  best = None
  for p in permutations(cost):
    t = sum(cost[r][g]
            for g, r in zip(tasks, p))
    if best is None or t < best[0]:
      best = (t, p)
  return best
```

The code works line by line as follows.

1. `best = None`: No assignment has been tried yet.
2. `for p in permutations(cost)`: `permutations` (from Python's `itertools` module) generates all orderings of the robots. `p = ("R3", "R1", "R2")` means that G1 is assigned to R3, G2 to R1, and G3 to R2.
3. `zip(tasks, p)`: Pairs the tasks with the robots in the ordering: (G1, R3), (G2, R1), (G3, R2).
4. `t = sum(cost[r][g] for g, r in ...)`: Computes the total time of this assignment.
5. `if best is None or t < best[0]`: If this is the first assignment or one better than any so far, `best` is updated. An assignment with an equal time does not replace the current best.

p	t	Comparison	best
(R1, R2, R3)	2 + 7 + 9 = 18	first assignment	(18, R1 R2 R3)
(R1, R3, R2)	2 + 8 + 4 = 14	14 < 18 ⇒ new best	(14, R1 R3 R2)

p	t	Comparison	best
(R2, R1, R3)	3 + 3 + 9 = 15	15 ≥ 14 ⇒ unchanged	(14, R1 R3 R2)
(R2, R3, R1)	3 + 8 + 5 = 16	16 ≥ 14 ⇒ unchanged	(14, R1 R3 R2)
(R3, R1, R2)	6 + 3 + 4 = 13	13 < 14 ⇒ new best	(13, R3 R1 R2)
(R3, R2, R1)	6 + 7 + 5 = 18	18 ≥ 13 ⇒ unchanged	(13, R3 R1 R2)

Table 10.9: Trace of `optimum()`. p lists the robots that take G1, G2, G3, respectively.

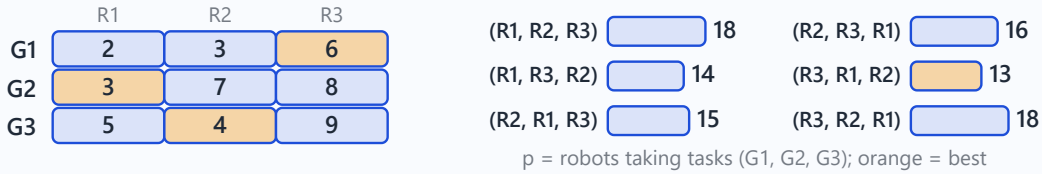


Figure 10.10: The end of the `optimum()` trace. On the right are the total times of the six orderings; the best is (R3, R1, R2) with 13 min (orange). The orange cells on the left are the optimal assignment.

Table 10.9 and Figure 10.10 show the result: the optimum is 13 minutes and is obtained with the assignment G1→R3, G2→R1, G3→R2. The assignment found by the contract net is the first row of the list (18 min). One more thing is visible in the table: two of the six assignments take 18 minutes, so in this example the contract net has found the worst possible total.

The problem with this method is scale. For n robots there are *n!* orderings, and this number grows very quickly. Table 10.10 shows a few values.

n	n! (number of assignments)	n^3
3	6	27
5	120	125
10	3,628,800	1,000
15	1,307,674,368,000	3,375

Table 10.10: The cost of trying all assignments ($n!$) versus the approximate number of steps of the Hungarian algorithm (n^3).

At n = 10 there are more than three and a half million assignments; at n = 15 the number exceeds a trillion. Trying all assignments is impractical except for small examples. The Hungarian algorithm finds the same optimum in $O(n^3)$ time: about a thousand steps at n = 10 and about three thousand at n = 15. But whichever algorithm is used, the real obstacle of the centralized solution is not computation but information: the center must know all the costs.

10.3.6 Bids and honesty

So far it has been assumed that the robots bid their costs honestly. Is this assumption valid if the robots are selfish agents? The answer depends on what the winner is paid. In the contract net, bidding is a reverse auction: the lowest bid wins, and the winner receives a payment for doing the job. The robot's gain is the difference between the payment it receives and its true cost.

Example 10.5

Consider the tender for G1. The true costs are R1 2, R2 3, R3 6. Two payment rules are compared.

1. **The winner is paid its own bid (first price).** If R1 bids 2 honestly, it wins, receives 2, and since its cost is 2 its gain is $2 - 2 = 0$.
2. **Inflating under first price.** Let R1 raise its bid to 2.9. If the other bids do not change, R1 still has the lowest bid (2.9 < 3), wins, and receives 2.9. Its gain is $2.9 - 2 = 0.9$. Under this rule the contractor wants to inflate its cost; an honest bid is not the best.
3. **The winner is paid the second-lowest bid (Vickrey).** R1 bids 2 honestly and wins; the payment is the second-lowest bid, 3. Its gain is $3 - 2 = 1$.
4. **Inflating does not help under Vickrey.** With any bid below 3 (2.5 or 2.9), R1 still wins and still receives 3: the gain of 1 does not change. If the bid goes above 3, R1 loses and its gain drops to 0. Inflating the bid either leaves the gain unchanged or lowers it.
5. **Underbidding does not help under Vickrey either.** R2's cost is 3. Suppose R2 bids 1.5 to get the job. It wins, but the payment is the second-lowest bid, R1's 2. Its gain is $2 - 3 = -1$: it makes a loss. With the honest bid (3) it loses the auction and its gain is 0; this is better.

Example 10.5 illustrates a general result. If the winner is paid its own bid, the contractor wants to inflate its cost. If the second-lowest bid is paid (the Vickrey rule), **an honest bid is best**: the robot's bid determines only whether it wins, not the payment. Because the payment depends on the others' bids, the robot has no way of raising the payment by lying. All the robot can do is decide whether winning is profitable, and an honest bid always makes that decision correctly. This result holds for a single task: if tasks are tendered in sequence and a robot that takes a task leaves later tenders, then because of the opportunity cost of later tasks an honest bid may no longer be dominant. The VCG mechanism, which chooses the whole assignment together, removes this problem.

This result is valid for the tender of a single task. When tasks are tendered in sequence, a robot may want to lose a tender on purpose in order to take a later task. Honesty for the assignment as a whole is secured by a general rule that makes each robot pay for the loss it imposes on the others: the **VCG mechanism** (Vickrey–Clarke–Groves). VCG is the multi-item generalization of the Vickrey rule.

10.3.7 Strengths, weaknesses, and improvements

The result in the example shows both the strength and the weakness of the contract net.

Strengths.

- **Distributed:** the manager makes a good assignment without knowing the cost table. The information stays with the robots and is shared only as far as necessary.
- **Few messages:** in a system with n robots, the number of messages per task is *O(n)*: one announcement (broadcast), at most n bids, and one award.
- **Flexibility:** adding a new robot is easy; the robot simply starts listening to announcements. The task of a robot that breaks down is put out to tender again.

Weaknesses.

- **Greedy and sequential:** in the example, 18 min against an optimum of 13 min (38% more). R1, which took G1 cheaply, stayed busy although it was the best candidate for G2.
- **Strategic bidding:** robots can inflate their costs strategically (10.3.6).
- **Order dependence:** the order in which tasks are put out to tender changes the result.

Order dependence is seen most clearly by trying all six tender orders with the same cost table. Table 10.11 gives the results.

Tender order	1st tender	2nd tender	3rd tender	Total
G1, G2, G3	G1→R1 (2)	G2→R2 (7)	G3→R3 (9)	18
G1, G3, G2	G1→R1 (2)	G3→R2 (4)	G2→R3 (8)	14
G2, G1, G3	G2→R1 (3)	G1→R2 (3)	G3→R3 (9)	15
G2, G3, G1	G2→R1 (3)	G3→R2 (4)	G1→R3 (6)	13
G3, G1, G2	G3→R2 (4)	G1→R1 (2)	G2→R3 (8)	14
G3, G2, G1	G3→R2 (4)	G2→R1 (3)	G1→R3 (6)	13

Table 10.11: The result of six tender orders on the same cost table. Each cell lists the task, the winner, and the winning bid.

Two of the six orders reach the optimum (13), and one gives the worst result (18). The optimum is found in the orders where G1 is left to the end: G1 has several good candidates (R1 2, R2 3, R3 6), so putting it out to tender last costs little. By contrast, if G1 comes first, R1 grabs it, even though R1 is also the best candidate for G2. Spending the scarce robot in the first tender makes the later tenders more expensive.

The weaknesses of the contract net are reduced by three improvements.

Re-tendering (repairing assignments by swaps)
: After the first assignment, robots propose pairwise swaps. If two robots exchanging their tasks reduces the total time, the swap is made. Local repairs undo some of the greedy decisions.

Combinatorial auction
: Robots bid not on single tasks but on bundles of tasks. In this way a robot can reflect in its bid the link between tasks, such as "if I take G1 I cannot take G2".

Second-price or VCG payments
: The winner is paid the second-best bid (or, in general, the loss it imposes on the others). An honest bid becomes the best strategy (10.3.6).

Example 10.6

The contract net's 18-minute assignment (G1→R1, G2→R2, G3→R3) is improved by re-tendering.

1. **Task swap between R1 and R2.** Before the swap, R1 does G1 (2) and R2 does G2 (7): total 9. After the swap, R1 does G2 (3) and R2 does G1 (3): total 6. The swap saves 3 minutes. The new assignment is G1→R2, G2→R1, G3→R3; total $3 + 3 + 9 = 15$.
2. **Task swap between R2 and R3.** Before the swap, R2 does G1 (3) and R3 does G3 (9): total 12. After the swap, R2 does G3 (4) and R3 does G1 (6): total 10. The swap saves 2 minutes. The new assignment is G1→R3, G2→R1, G3→R2; total $6 + 3 + 4 = 13$.
3. **Is there another profitable swap?** 13 is the optimum (Table 10.9); no swap can lower the total further. The optimum is reached with two swaps.

Example 10.6 shows the power of re-tendering, but a caveat is needed: pairwise swaps may not reach the optimum in every instance. There are non-optimal assignments in which no pairwise swap yields a gain. Swapping is therefore a practical tool for improving the greedy result, not a guarantee of the optimum.

10.3.8 Common mistakes in task allocation

Common mistake

Assuming the manager knows the costs. In the contract net the manager does not see the cost table; it learns the costs only from the bids. If the manager knew the table there would be no need for a tender; it would compute the optimum directly.

Thinking a busy robot bids. Each robot takes at most one task. In the G2 tender R1's cost (3) is the lowest, but R1 does not bid because it is busy. The winner of G2 is R2 (7).

Looking for the cause of the loss in honesty. In the example the robots bid honestly and the result is still not optimal. The cause of the loss is not strategic bidding but the greedy, sequential tender. Inflating costs does not reduce this loss.

Overstating the number of messages. The number of messages per task is $O(n)$, not $O(n^2)$: one announcement, n bids, one award. Robots exchange messages only with the manager, not with one another.

Computing the percentage against the wrong base. 18 minutes is $(18 - 13)/13 \approx 38\%$ above the optimum. $(18 - 13)/18 \approx 28\%$ is the share of the contract net's time that is wasted; the two answer different questions.

10.4 Coalitional games

In task allocation each robot did its job alone. In many jobs, though, agents working together produce value that none of them could produce alone. This section studies the formation of groups and the sharing of the value a group produces among its members. This branch of game theory is called **cooperative game theory** or **coalitional games**. The branch has three basic tools: the Shapley value as a fairness criterion, the core as a stability criterion, and power indices that measure strength in voting.

10.4.1 Coalition formation

Agents that cannot reach a goal alone can form temporary or permanent **coalitions**. Freight companies share vehicles: instead of two half-empty trucks, one full truck is sent. Drones carry together a heavy load that none could lift alone. Computing servers split a job into pieces and solve it together. In every case the group produces more than its members could produce alone.

Definition 10.6

A **coalition** is a group formed by agents that decide to act together. The coalition consisting of all agents is called the **grand coalition**. The process of deciding which coalitions the agents split into is called **coalition formation**.

These games differ from the games of the earlier sections at one point: they are **cooperative**. In the Prisoner's Dilemma, even if the players agreed to "both stay silent", the agreement had no enforcement; each player would have wanted to break it. In coalitional games it is assumed that **binding agreements** can be made: a contract, a law, or a software protocol makes the members of a coalition keep their promises. So the question changes. It is no longer "which strategy will each player choose?" but this: **who joins with whom, and how is the gain shared?**

The two questions are linked. If the sharing is not fair the coalition falls apart: agents who feel wronged leave and form another group. The stability of a coalition depends on the sharing rule. For this reason fairness and stability are treated together.

The Shapley value

A **fairness** criterion: everyone receives their average marginal contribution (10.4.5).
The core

A **stability** criterion: no subgroup can secede and get more (10.4.9).

10.4.2 Coalition structures

Agents do not have to unite in a single large group. Consider three logistics firms A, B, and C. The three can work separately, two can merge while the third stays apart, or all three can join together.

Definition 10.7

A **coalition structure** is a partition of the agents into disjoint coalitions. Every agent is in exactly one coalition, and no two coalitions share a member. The **total value** of a structure is the sum of the values of the coalitions in it.

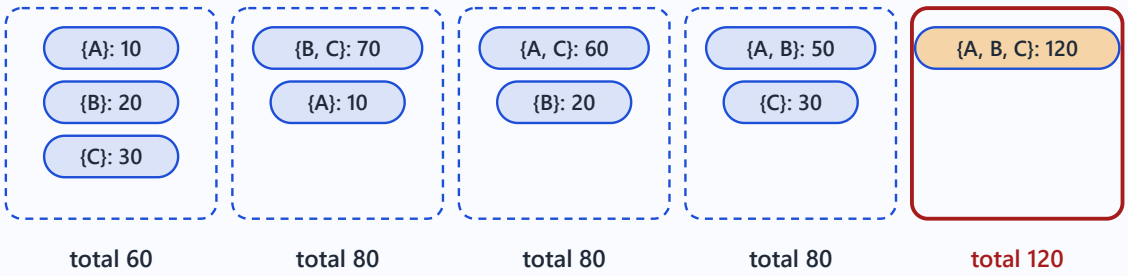


Figure 10.11: The five coalition structures of three firms and their total values. Alone, the firms produce 60 in total; one pair plus a single firm produces 80; the grand coalition produces 120. The best structure is the grand coalition (red frame).

Figure 10.11 shows all structures of the three agents. The values come from the characteristic function given in Table 10.12. If everyone works alone the total is $10 + 20 + 30 = 60$. If {B, C} merge and A stays apart, the total is $70 + 10 = 80$. {A, C} with {B}: $60 + 20 = 80$. {A, B} with {C}: $50 + 30 = 80$. All three together produce 120. From society's point of view the question is: **which structure has the largest total value?** In this example the answer is the grand coalition.

Number of structures. The number of ways to partition n agents into non-empty disjoint groups is called the **Bell number** and is denoted B_n. For three agents there are 5 structures. Bell numbers can be computed step by step with the following relation: count according to how many of the first n agents are in the group that contains agent n + 1. If this group contains k of the first n agents, there are $C(n, k)$ ways to choose those k agents, and the remaining $n - k$ agents are partitioned in B_{n-k} ways. The total is:

$$B_{n+1} = \Sigma_{k=0}^{n} C(n, k) \cdot B_{n-k},\ B_0 = 1.$$

With this relation we get $B_1 = 1$, $B_2 = 1 + 1 = 2$, $B_3 = B_2 + 2B_1 + B_0 = 2 + 2 + 1 = 5$. For four agents, $B_4 = C(3,0)\cdot B_3 + C(3,1)\cdot B_2 + C(3,2)\cdot B_1 + C(3,3)\cdot B_0 = 5 + 6 + 3 + 1 = 15$. The number grows very quickly: there are 4,140 structures for n = 8 and 115,975 for n = 10. In general, finding the best structure is **NP-hard** (Sandholm et al., 1999); that is, as the number of agents grows, no known algorithm solves it quickly on every instance.

In the example game this difficulty does not arise, because the game is **superadditive**: joining together never hurts. In this case the best structure is always the grand coalition, and only one question remains: how should the 120 produced by the grand coalition be shared?

10.4.3 The characteristic function

Definition 10.8

Let $N = \{1, \ldots, n\}$ be the set of players. A **coalitional game** is given by a function that assigns a value $v(S)$ to every coalition $S \subseteq N$. $v(S)$ is the total value the members of S can obtain together, independently of the others. The value of the empty coalition is 0: $v(\emptyset) = 0$. The function v is called the **characteristic function**.

If the gain can be distributed freely within the group, that is, if the value of the coalition can be divided like money and handed out among the members in any desired proportion, such games are called **transferable-utility games** (TU games).

The characteristic function gathers all the information about the game in a single table. Which actions the players choose and how they work do not show in the table; only how much value each group ends up producing does. This abstraction is the foundation of cooperative game theory: since the details inside a group are settled by binding agreements, only the strength of the groups matters.

The main example of the chapter is three logistics firms. Together the firms make a profit of 120 thousand TL; the best pair makes 70 and the best single firm 30. All values of the characteristic function are given in Table 10.12.

S	{A}	{B}	{C}	{A, B}	{A, C}	{B, C}	{A, B, C}
v(S)	10	20	30	50	60	70	120

Table 10.12: The characteristic function $v(S)$ of the three logistics firms (profit, thousand TL).

Definition 10.9

A coalitional game is **superadditive** if, for every pair of disjoint coalitions S and T ($S \cap T = \emptyset$),

$$v(S \cup T) \geq v(S) + v(T)$$

holds. The merger of two disjoint groups produces no less than the sum of what they produce separately; that is, joining together does no harm.

Example 10.7

That the logistics game is superadditive is shown by checking all pairs of disjoint coalitions. Pairs that involve the empty coalition hold automatically ($v(\emptyset) = 0$). Six pairs remain.

1. **Two singletons merge.** $v(AB) = 50 \geq 10 + 20 = 30$ ✓; $v(AC) = 60 \geq 10 + 30 = 40$ ✓; $v(BC) = 70 \geq 20 + 30 = 50$ ✓.
2. **A pair and a singleton merge.** $v(ABC) = 120 \geq v(BC) + v(A) = 70 + 10 = 80$ ✓; $120 \geq v(AC) + v(B) = 60 + 20 = 80$ ✓; $120 \geq v(AB) + v(C) = 50 + 30 = 80$ ✓.

All six inequalities hold; the game is superadditive. For this reason the grand coalition {A, B, C} makes the total value largest, and the value to be shared is 120. Whether the coalition forms and survives, however, depends on how this value is shared: even in a superadditive game the core can be empty.

Superadditivity does not hold in every game. As a coalition grows, the cost of coordination can rise: vehicle planning becomes harder in a very large freight partnership. In such a game two separate groups can produce more than the merged group, and the best structure need not be the grand coalition.

10.4.4 How should 120 be split?

The grand coalition produces 120. How should this value be shared? There are two conditions that every reasonable split must satisfy.

Definition 10.10

An **imputation** $x = (x_A, x_B, x_C)$ is a payoff vector that satisfies two conditions:

- **Efficiency:** its sum equals the value of the grand coalition, $x_A + x_B + x_C = v(N) = 120$. No value is wasted.
- **Individual rationality:** everyone gets at least their stand-alone value, $x_i \geq v(\{i\})$. Otherwise that agent would leave the coalition and work alone.

There are infinitely many imputations that satisfy these two conditions. Table 10.13 compares four proposals.

Proposal	x_A	x_B	x_C	Rationale / problem
Equal split	40	40	40	Simple; but ignores C's larger contribution
Proportional to stand-alone value (10 : 20 : 30)	20	40	60	Looks only at $v(\{i\})$; ignores synergies
The strong C's proposal	10	25	85	A and B together can get 50; they will not accept
Shapley value	30	40	50	Average marginal contribution

Table 10.13: Four proposals for splitting 120.

All four proposals are imputations: they sum to 120 and everyone gets at least their stand-alone value. In C's proposal, for example, A gets exactly 10, so it does not fall below its stand-alone value. Even so, the proposals are very different.

The **equal split** is the simplest rule, but it ignores that C adds more value to every group. The **proportional split** divides 120 in proportion to the stand-alone values: since the total stand-alone value is $10 + 20 + 30 = 60$, each player gets $120/60 = 2$ times its stand-alone value. This rule does not see synergies: it does not account for the extra value of 20 that A creates with B and 20 with C ($50 - 30$ and $60 - 40$). **The strong C's proposal** gives C 85. But then A and B together get $10 + 25 = 35$, whereas if the two broke away and formed their own coalition they would earn $v(AB) = 50$. This proposal will not be accepted. The **Shapley value**, in contrast, gives each player the average of the value it adds to all possible groups.

The table brings out two separate criteria: is the split **fair** (the Shapley value) and is it **stable** (the core)? C's proposal is not stable, because a subgroup wants to leave. The equal split is stable in this game, but it does not reflect contributions.

10.4.5 The Shapley value: definition and axioms

The idea of the Shapley value is explained with a simple story. The players arrive one by one in a random order and join the coalition. Each player is credited with the value it adds to the coalition at the moment it arrives. For example, if the order is A → B → C, A arrives first and adds $v(A) - v(\emptyset) = 10$

to the empty coalition. When B arrives, the coalition grows from {A} to {A, B}, and B's contribution is $v(AB) - v(A) = 50 - 10 = 40$. C arrives last and adds $v(ABC) - v(AB) = 120 - 50 = 70$. This value that a player adds to a coalition is called its **marginal contribution**.

A single order is not fair: the player who arrives last often collects the largest synergy. This is why the Shapley value averages over all orders.

Definition 10.11

The **Shapley value** φ_i of player i is the average marginal contribution of i over all $n!$ arrival orders of the n players:

$$\varphi_i = (1/n!) \cdot \Sigma_\pi \left[v(P_i^\pi \cup \{i\}) - v(P_i^\pi) \right]$$

Here the sum runs over all orders π, and P_i^π is the set of players that come before i in the order π.

The same value can also be written over coalitions instead of orders. The number of orders in which the set of players before i is S equals the number of arrangements of the members of S among themselves ($|S|!$) times the number of arrangements of those after i ($(n - |S| - 1)!$). Therefore

$$\varphi_i = \Sigma_{S \subseteq N \setminus \{i\}} \left[|S|! \cdot (n - |S| - 1)! / n! \right] \cdot \left[v(S \cup \{i\}) - v(S) \right]$$

holds. This formula sums over the 2^{n-1} coalitions that do not contain i, instead of over $n!$ orders. In a three-player game the weights are as follows: for $|S| = 0$, $0! \cdot 2!/3! = 2/6 = 1/3$; for $|S| = 1$, $1! \cdot 1!/3! = 1/6$; for $|S| = 2$, $2! \cdot 0!/3! = 2/6 = 1/3$. The weights add up to $1/3 + 2 \cdot 1/6 + 1/3 = 1$.

Computational cost. The exact computation requires either $n!$ orders or 2^{n-1} subsets. Both grow very quickly as n grows. For large n the Shapley value is computed approximately by **sampling**: a few thousand random orders are chosen and the marginal contributions in these orders are averaged. As the number of samples grows, the average approaches the true value.

The importance of the Shapley value does not come from its being merely an intuitive rule. Shapley (1953) showed that this value is the only rule that satisfies four natural conditions. These conditions are called **axioms**.

Theorem 10.1 (Shapley, 1953)

Among the rules that assign a payoff vector to every coalitional game, the only rule that satisfies the following four axioms is the Shapley value.

- **Efficiency:** $\Sigma_i \varphi_i = v(N)$. The value of the grand coalition is distributed exactly.
- **Symmetry:** Two players that make the same contribution receive the same share. Formally, if $v(S \cup \{i\}) = v(S \cup \{j\})$ for every S containing neither i nor j, then $\varphi_i = \varphi_j$.
- **Dummy player:** A player that contributes nothing to any coalition, that is, a player with $v(S \cup \{i\}) = v(S)$ for every S, receives $\varphi_i = 0$.
- **Additivity:** If the values of two games are added, the Shapley shares add as well: $\varphi_i(v + w) = \varphi_i(v) + \varphi_i(w)$.

The proof of the theorem is outside the scope of these notes. But it is easy to see why the definition satisfies the axioms. **Efficiency:** in every order, the sum of the marginal contributions is the sum of the values added step by step from the empty coalition to the grand coalition, and it telescopes to $v(N) - v(\emptyset) = v(N)$. Averaging leaves the total at $v(N)$. **Dummy player:** a player whose contribution is 0 in every order has an average of 0. **Symmetry:** a mapping that swaps i and j matches i's contribution in each order with j's contribution in another order; the averages come out equal. **Additivity:** the marginal contribution is linear in v; the contribution of a sum is the sum of the contributions.

The Shapley value is used today outside game theory as well. In machine learning, the **SHAP** (SHapley Additive exPlanations) method, which splits a single decision of a model into the contributions of the input features, treats the features as "players" and the model's prediction as the "coalition value", and assigns each feature its Shapley value.

10.4.6 The Shapley value step by step

In the logistics game the Shapley value is computed by writing out the marginal contributions in each of the six arrival orders.

Example 10.8

In each order the agents arrive one by one; for each, the value it adds to the coalition at the moment it arrives is written down. The values are read from Table 10.12.

1. **A → B → C.** A: $10 - 0 = 10$; B: $v(AB) - v(A) = 50 - 10 = 40$; C: $v(ABC) - v(AB) = 120 - 50 = 70$. Row total $10 + 40 + 70 = 120$.
2. **A → C → B.** A: $10 - 0 = 10$; C: $v(AC) - v(A) = 60 - 10 = 50$; B: $v(ABC) - v(AC) = 120 - 60 = 60$. Row total 120.
3. **B → A → C.** B: $20 - 0 = 20$; A: $v(AB) - v(B) = 50 - 20 = 30$; C: $120 - 50 = 70$. Row total 120.

4. **B → C → A.** B: *20*; C: *v(BC) − v(B) = 70 − 20 = 50*; A: *v(ABC) − v(BC) = 120 − 70 = 50*. Row total 120.
5. **C → A → B.** C: *30 − 0 = 30*; A: *v(AC) − v(C) = 60 − 30 = 30*; B: *120 − 60 = 60*. Row total 120.
6. **C → B → A.** C: *30*; B: *v(BC) − v(C) = 70 − 30 = 40*; A: *120 − 70 = 50*. Row total 120.
7. **Add up the columns and divide by 6.** A: *10 + 10 + 30 + 50 + 30 + 50 = 180*, $\varphi_A = 180/6 = 30$. B: *40 + 60 + 20 + 20 + 60 + 40 = 240*, $\varphi_B = 40$. C: *70 + 50 + 70 + 50 + 30 + 30 = 300*, $\varphi_C = 50$.

The Shapley value is $\varphi = (30, 40, 50)$. Its sum is 120; the efficiency axiom holds.

Order	A	B	C
A → B → C	10	40	70
A → C → B	10	60	50
B → A → C	30	20	70
B → C → A	50	20	50
C → A → B	30	60	30
C → B → A	50	40	30
Total	180	240	300
φ = total / 6	30	40	50

Table 10.14: Marginal contributions in the six arrival orders. Each row sums to *v(ABC) = 120*.

Table 10.14 shows the computation at a glance. Two regularities in the table help check the computation. First, every row sums to 120; if a row does not total 120, there is a subtraction error in it. Second, in the orders where a player arrives first, its contribution is always its stand-alone value: A's first two rows are 10, B's third and fourth rows are 20, and C's last two rows are 30.

Checking with the coalition formula. The same result is obtained with the weighted formula of 10.4.5. For A there are four coalitions that do not contain A: ∅ (weight 1/3), {B} and {C} (weight 1/6), and {B, C} (weight 1/3).

$$\varphi_A = (1/3)\cdot 10 + (1/6)\cdot(50 - 20) + (1/6)\cdot(60 - 30) + (1/3)\cdot(120 - 70) = 10/3 + 5 + 5 + 50/3 = 30$$

For B: $(1/3)\cdot 20 + (1/6)\cdot(50 - 10) + (1/6)\cdot(70 - 30) + (1/3)\cdot(120 - 60) = 20/3 + 20/3 + 20/3 + 20 = 40$. For C: $(1/3)\cdot 30 + (1/6)\cdot(60 - 10) + (1/6)\cdot(70 - 20) + (1/3)\cdot(120 - 50) = 10 + 25/3 + 25/3 + 70/3 = 50$. The two methods give the same result.

Interpreting the result. C receives the largest share (50), because it is both the most valuable player alone (30) and the one that adds the most value to every group. A receives the smallest share (30). Still, A's share is three times its stand-alone value (10), because the synergy A adds to groups is large: A adds 50 to {B, C}. The Shapley value gives each player its stand-alone value plus its average share of the synergies it creates.

10.4.7 Code trace: shapley()

The definition of the Shapley value can be turned into code directly. The function takes two inputs: the player list `N = ["A", "B", "C"]` and the characteristic function `v`. `v` is a dictionary whose keys are `frozenset`s: `v[frozenset()] = 0`, `v[frozenset({"A"})] = 10`, …, `v[frozenset({"A", "B", "C"})] = 120`. A `frozenset` is an immutable set; an ordinary set cannot be a dictionary key, but an immutable one can.

```
from itertools import permutations
def shapley(N, v):
  phi = {i: 0 for i in N}
  orders = list(permutations(N))
  for order in orders:
    S = frozenset()
    for i in order:
      gain = v[S | {i}] - v[S]
      phi[i] += gain
      S = S | {i}
  return {i: phi[i] / len(orders) for i in N}
```

The code works line by line as follows.

1. `phi = {i: 0 for i in N}`: Each player's total contribution starts at 0.
2. `orders = list(permutations(N))`: `permutations` generates all arrival orders: 3! = 6 orders for three players.
3. `for order in orders`: Each order is processed separately. With `S = frozenset()` the coalition starts empty.
4. `for i in order`: The players arrive in the sequence of the order.
5. `gain = v[S | {i}] - v[S]`: The marginal contribution of i. `S | {i}` is S with i added; since the union of a `frozenset` and an ordinary set is again a `frozenset`, it can be used as a dictionary key.
6. `phi[i] += gain` and `S = S | {i}`: The gain is added to i's total, and i joins the coalition.
7. `return {i: phi[i] / len(orders) for i in N}`: The totals are divided by the number of orders (6).

order	i	S (before)	gain	phi (A, B, C)
(A, B, C)	A	∅	10 − 0 = 10	(10, 0, 0)
	B	{A}	50 − 10 = 40	(10, 40, 0)
	C	{A, B}	120 − 50 = 70	(10, 40, 70)
(A, C, B)	A	∅	10 − 0 = 10	(20, 40, 70)
	C	{A}	60 − 10 = 50	(20, 40, 120)
	B	{A, C}	120 − 60 = 60	(20, 100, 120)
(B, A, C)	B	∅	20 − 0 = 20	(20, 120, 120)
	A	{B}	50 − 20 = 30	(50, 120, 120)
	C	{A, B}	120 − 50 = 70	(50, 120, 190)
(B, C, A)	B	∅	20 − 0 = 20	(50, 140, 190)
	C	{B}	70 − 20 = 50	(50, 140, 240)
	A	{B, C}	120 − 70 = 50	(100, 140, 240)
(C, A, B)	C	∅	30 − 0 = 30	(100, 140, 270)
	A	{C}	60 − 30 = 30	(130, 140, 270)
	B	{A, C}	120 − 60 = 60	(130, 200, 270)
(C, B, A)	C	∅	30 − 0 = 30	(130, 200, 300)
	B	{C}	70 − 30 = 40	(130, 240, 300)
	A	{B, C}	120 − 70 = 50	(180, 240, 300)

Table 10.15: Trace of `shapley(N, v)`. Each row is one step of the inner loop; the last column is `phi` after that step.

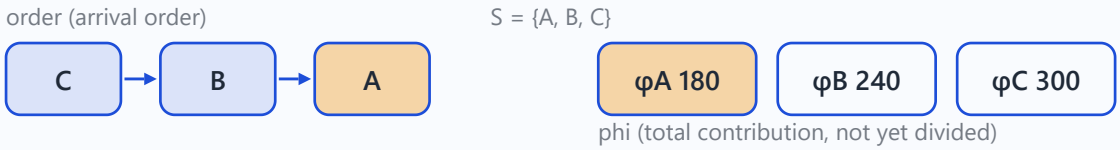


Figure 10.12: The last step of the last order in the `shapley()` trace. The order is C → B → A, and A joins last (orange); the `phi` totals (180, 240, 300) have not yet been divided by 6.

Table 10.15 and Figure 10.12 show the trace. When the loops finish, `phi = {A: 180, B: 240, C: 300}`. The last line divides each total by 6 and returns `{A: 30.0, B: 40.0, C: 50.0}`. The trace accumulates the columns of Table 10.14 row by row; at the end of each order the sum of `phi` has grown by 120 (120 after the first order, 240 after the second, ..., 720 = 6 · 120 after the sixth).

The cost of the code is $n! \cdot n$ steps: n players are processed in each of the n! orders. For n = 10 this is about 36 million steps and is still feasible for a computer; for n = 20 it is $20! \cdot 20 \approx 4.9 \cdot 10^{19}$ steps and is not. For large games, a few thousand randomly chosen orders are used in place of the `orders` list; the rest of the code does not change.

10.4.8 Back to the game: the pairwise synergy model

Let us return to the opening Shapley Coalition Game (10.1.1). In the game, the value of a coalition is the sum of its members' stand-alone values plus the synergy bonus of every pair inside it. In the sample round the stand-alone values are A 6, B 4, C 5, and the synergies are A–B 4, A–C 2, B–C 0. The characteristic function is: $v(A) = 6$, $v(B) = 4$, $v(C) = 5$, $v(AB) = 6 + 4 + 4 = 14$, $v(AC) = 6 + 5 + 2 = 13$, $v(BC) = 4 + 5 + 0 = 9$, $v(ABC) = 15 + 6 = 21$.

In this model there is no need to compute the six orders one by one; the Shapley value reduces to a short rule.

Theorem 10.2 (The Shapley value in the pairwise synergy model)

Let each player's stand-alone value be a_i, each pair's synergy be s_{ij}, and the value of a coalition be $v(S) = \Sigma_{i \in S} a_i + \Sigma_{\{i,j\} \subseteq S} s_{ij}$. Then

$$\varphi_i = a_i + \tfrac{1}{2} \cdot \Sigma_{j \neq i} s_{ij}$$

That is, each player receives its stand-alone value and half of every synergy it takes part in.

Proof. The proof uses the Shapley axioms. The game is written as a sum of simpler games. For each i define a "stand-alone game": $u_i(S) = a_i$ if i ∈ S, and 0 otherwise. For each pair {i, j} define a "synergy game": $w_{ij}(S) = s_{ij}$ if i and j are both in S, and 0 otherwise. From the definition, $v = \Sigma_i u_i + \Sigma_{\{i,j\}} w_{ij}$.

In the stand-alone game u_i, every player other than i is a dummy: their presence does not change the value of any coalition. By the dummy player axiom they get 0; by the efficiency axiom, i receives all of $u_i(N) = a_i$.

In the synergy game w_{ij}, the players other than i and j are dummies and get 0. i and j are symmetric: $w_{ij}(S \cup \{i\}) = w_{ij}(S \cup \{j\}) = 0$ for every S containing neither i nor j. By the symmetry axiom they receive equal shares; since by efficiency these shares sum to s_{ij}, each receives $s_{ij}/2$.

By the additivity axiom, the Shapley value of v is the sum of the Shapley values of these simple games. Player i receives a_i from its own stand-alone game and $s_{ij}/2$ from every synergy game it takes part in. □

Player	Alone	Synergy share	φ	Equal share
A	6	4/2 + 2/2 = 3	9	7
B	4	4/2 + 0 = 2	6	7
C	5	2/2 + 0 = 1	6	7
Total	15	6	21	21

Table 10.16: The Shapley value and the equal split in the sample round of the game.

Example 10.9

The result $\varphi = (9, 6, 6)$ given by the rule is checked with the six arrival orders.

1. **A → B → C:** A 6, B $14 - 6 = 8$, C $21 - 14 = 7$.
2. **A → C → B:** A 6, C $13 - 6 = 7$, B $21 - 13 = 8$.
3. **B → A → C:** B 4, A $14 - 4 = 10$, C 7.
4. **B → C → A:** B 4, C $9 - 4 = 5$, A $21 - 9 = 12$.
5. **C → A → B:** C 5, A $13 - 5 = 8$, B 8.
6. **C → B → A:** C 5, B $9 - 5 = 4$, A 12.
7. **Average.** A: $(6 + 6 + 10 + 12 + 8 + 12)/6 = 54/6 = 9$. B: $(8 + 8 + 4 + 4 + 8 + 4)/6 = 36/6 = 6$. C: $(7 + 7 + 7 + 5 + 5 + 5)/6 = 36/6 = 6$.

The average over the six arrival orders also gives $(9, 6, 6)$.

Now the two opening questions can be answered. **Does the equal split treat A unfairly?** Measured by contribution, yes: A's Shapley share is 9, while the equal split gives 7. A falls 2 short of its contribution, while B and C each receive 1 more than theirs. **Should the most valuable player alone always receive the largest share?** In this round A is both the most valuable player alone and the player with the largest share, but that is a coincidence. The share is found by adding half of the synergies to the stand-alone value. A player with a low stand-alone value but a partner in large synergies can receive more than a player with a high stand-alone value but no synergy with anyone. For example, if the synergies were A–B 10, A–C 0, B–C 0, B's share would be $4 + 5 = 9$ and C's would be 5; C, with the higher stand-alone value, would receive less.

The game measures the outcome by an **equal-split error**: the total deviation of the equal split from the Shapley value. In the sample round this error is $|7 - 9| + |7 - 6| + |7 - 6| = 2 + 1 + 1 = 4$.

How do people actually split? Experiments show that the theory alone is not sufficient. In three-person coalitional games, the choices of subjects who propose a "fair" split for others are best explained as a mixture of the Shapley value and the equal split (de Clippel and Rozen, 2022). People care about contribution, but they also care about equality. In the example, a fifty-fifty mixture gives $\frac{1}{2} \cdot (9, 6, 6) + \frac{1}{2} \cdot (7, 7, 7) = (8, 6.5, 6.5)$. Splitting by contribution (Shapley) and splitting equally often diverge; which one counts as fair depends on the context. In a partnership, investors may find a split by contribution fair, while in a family siblings may find an equal split fair.

10.4.9 The core

The Shapley value is a fairness criterion: it gives each player the average of its contributions. But a fair split need not be stable. If a subgroup gets more by seceding from the grand coalition and sharing its own value, that group leaves, however fair the split may look. The notion that measures stability is the core.

Definition 10.12

A coalition S **blocks** an imputation x if the total its members receive in x is less than $v(S)$, that is, if $\Sigma_{i \in S} x_i < v(S)$: the members of S can leave and, by sharing $v(S)$ among themselves, all be better off. An imputation x is **in the core** if and only if

$$\Sigma_{i \in N} x_i = v(N) \text{ and } \Sigma_{i \in S} x_i \geq v(S) \text{ for every } S \subseteq N$$

hold; that is, if no coalition blocks x.

The idea of "blocking" becomes concrete as follows. If the members of S receive a total of 35 in x, but can earn 50 together if they leave, they can split the 50 so that each gets more than in x (for example by splitting the extra 15 equally). So S has no reason to accept x. The core is the set of imputations against which no group has such an objection.

The inequalities in the definition include individual rationality for one-player S's and efficiency for the grand coalition. Therefore every payoff vector in the core is an imputation. In the logistics game, the constraints that define the core are:

Singleton constraints

$$x_A \geq 10, x_B \geq 20, x_C \geq 30$$

Pair constraints

$$x_A + x_B \geq 50, x_A + x_C \geq 60, x_B + x_C \geq 70$$

Efficiency

$$x_A + x_B + x_C = 120$$

The core is a **system of linear inequalities** (Gillies, 1959). The solution set of linear inequalities is a polygonal (in general, polyhedral) region. So the core can be empty, can consist of a single point, or can be a region. An empty core and a region-shaped core are both seen with examples in this section.

A shortcut for the pair constraints. Using the efficiency equation, the pair constraints turn into upper bounds on single players. Since $x_A + x_B \geq 50$ and $x_A + x_B = 120 - x_C$, we get $120 - x_C \geq 50$, that is, $x_C \leq 70$. In the same way $x_B \leq 120 - 60 = 60$ and $x_A \leq 120 - 70 = 50$. In general, in a three-player game each player can receive at most $v(N) - v(N \setminus \{i\})$. This value is what i adds to the grand coalition as the last one to join; if i receives more than this, the other two leave and take their own value.

The Shapley value and the core. The Shapley value is a fairness criterion and the core a stability criterion; the two do not always coincide. The Shapley value can lie outside the core, and when the core is empty the Shapley value still exists. But in one class of games the two are compatible.

Definition 10.13

A coalitional game is **convex** if each player's marginal contribution does not decrease as the coalition it joins grows: for every i and $S \subseteq T \subseteq N \setminus \{i\}$, $v(S \cup \{i\}) - v(S) \le v(T \cup \{i\}) - v(T)$. A convex game has a kind of "snowball effect": as the group grows, the value a newcomer adds increases or stays the same.

Theorem 10.3 (Shapley, 1971)

In a convex game the core is non-empty, and the Shapley value is always in the core.

The proof of the theorem is long, but it is easy to check that the logistics game is convex. Table 10.17 lists the value each player adds to four coalitions. In each row the values do not decrease from left to right: as the coalition grows the contribution does not shrink. Our example is a convex game; therefore the Shapley split $(30, 40, 50)$ is in the core. This is also confirmed by the constraint check in 10.4.11.

Player	to ∅	to a single player	to a single player	to a pair
A	10	{B}: 30	{C}: 30	{B, C}: 50
B	20	{A}: 40	{C}: 40	{A, C}: 60
C	30	{A}: 50	{B}: 50	{A, B}: 70

Table 10.17: Marginal contributions in the logistics game. The contribution does not decrease as the coalition grows; the game is convex.

Convexity is a sufficient condition, not a necessary one: in a non-convex game too, the Shapley value can be in the core. The condition that decides exactly whether the core is empty is the **Bondareva–Shapley theorem** (Bondareva, 1963; Shapley, 1967): the core is non-empty if and only if the game is **balanced**. Although the general definition is technical, for three-player games the condition becomes concrete. In a three-player game the core is non-empty if and only if the following two conditions hold:

- The total value of every coalition structure does not exceed $v(N)$: $v(N) \ge v(A) + v(B) + v(C)$ and, for each i, $v(N) \ge v(\{i\}) + v(N \setminus \{i\})$.
- Half the sum of the values of the pairs does not exceed $v(N)$: $v(N) \ge \tfrac{1}{2} \cdot [v(AB) + v(AC) + v(BC)]$.

Why the second condition is necessary is easy to see: for an x in the core, adding the three pair constraints gives $2(x_A + x_B + x_C) \ge v(AB) + v(AC) + v(BC)$; the left side is $2v(N)$. In the logistics game $\tfrac{1}{2} \cdot (50 + 60 + 70) = 90 \le 120$ and the value of every structure is at most 80; both conditions hold.

10.4.10 The geometry of the core

In a three-player game, imputations can be drawn on a triangle. Thanks to the efficiency equation $x_A + x_B + x_C = 120$, once two of the three numbers are chosen the third is determined; so the imputations form a two-dimensional region.

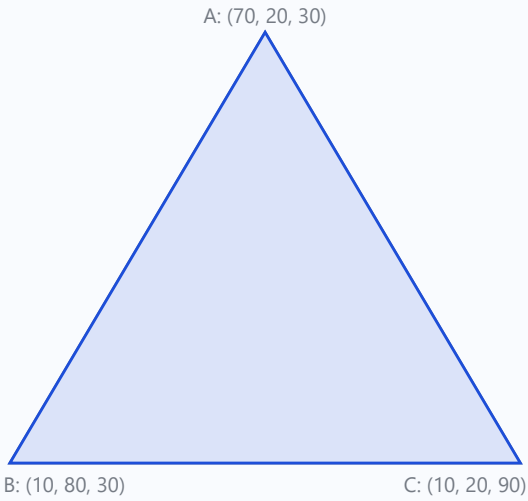


Figure 10.13: All individually rational and efficient imputations form a triangle. At each vertex one player receives all of the 60 that lies above the stand-alone values: vertex A (70, 20, 30), vertex B (10, 80, 30), vertex C (10, 20, 90).

Figure 10.13 shows the starting situation. Each player first receives its stand-alone value: $10 + 20 + 30 = 60$. A surplus of $120 - 60 = 60$ units remains. Each way of dividing this surplus among the three players is a point of the triangle. At the vertices the entire surplus goes to a single player. For example, at vertex C, C receives all 60 in addition to its own 30, giving the imputation $(10, 20, 90)$.

The core is found by cutting from this triangle the pieces that the pair constraints slice off.

1. **The {A, B} constraint.** $x_A + x_B \geq 50 \Leftrightarrow x_C \leq 70$. If C is given more than 70, A and B together get less than 50; they leave and take their own 50. This constraint cuts off the small triangle around vertex C.
2. **The {A, C} constraint.** $x_A + x_C \geq 60 \Leftrightarrow x_B \leq 60$. Vertex B is cut off.
3. **The {B, C} constraint.** $x_B + x_C \geq 70 \Leftrightarrow x_A \leq 50$. Vertex A is cut off too.

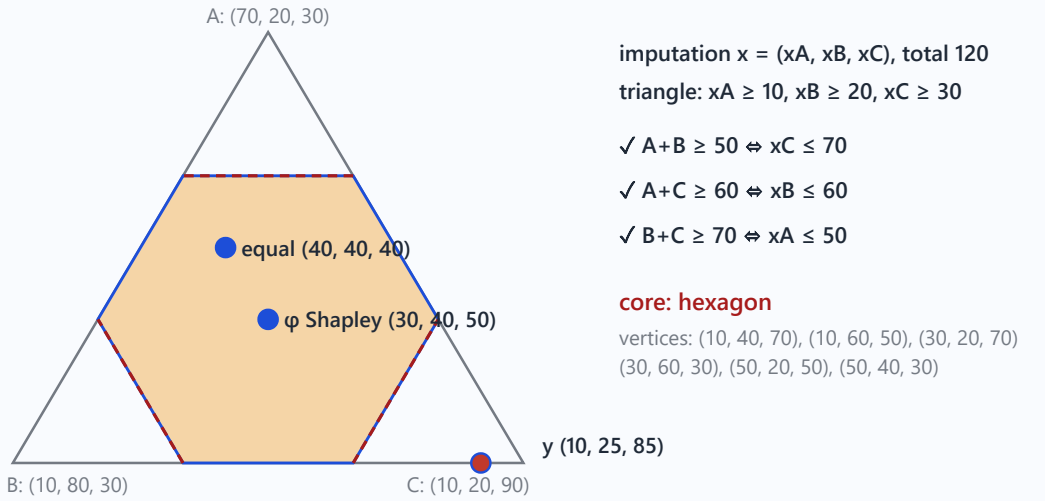


Figure 10.14: The orange hexagon left after the three vertices are cut off is the core. The Shapley value $\varphi = (30, 40, 50)$ and the equal split $(40, 40, 40)$ are in the core (blue points); $y = (10, 25, 85)$ is outside it (red point).

In Figure 10.14 the remaining region is a polygon with six vertices. Its vertices are the points where each player's share is at its lower bound (the stand-alone value) or its upper bound ($v(N) - v(N \setminus \{i\})$): $x_A \in [10, 50]$, $x_B \in [20, 60]$, $x_C \in [30, 70]$. If one player is at its lower bound and another at its upper bound, the third's share is fixed by efficiency. For example, if A gets 10 and B gets 60, C gets $120 - 70 = 50$. The six vertices are: $(10, 40, 70)$, $(10, 60, 50)$, $(30, 20, 70)$, $(30, 60, 30)$, $(50, 20, 50)$, $(50, 40, 30)$.

The three points in the figure show three different situations with respect to the core. The Shapley value is inside the hexagon: it does not touch any of the three boundaries. The equal split is inside as well: $40 \le 50$, $40 \le 60$, $40 \le 70$. C's proposal y, however, lies in the cut-off piece at vertex C: $x_C = 85 > 70$. The geometry also answers the question "which coalition blocks?": the edge on which y falls outside is the edge of the {A, B} constraint; the coalition that blocks y is {A, B}.

10.4.11 Checking the core step by step

Whether an imputation is in the core is determined by checking each of the seven constraints one by one. If an imputation is in the core, no coalition can leave, take its own value ($v(S)$), and be better off.

Example 10.10

The Shapley imputation $x = (30, 40, 50)$ and C's proposal $y = (10, 25, 85)$ are tested against the seven constraints.

1. **{A}: v(A) = 10.** For x, $30 \ge 10$ ✓; for y, $10 \ge 10$ ✓ (equality suffices).
2. **{B}: v(B) = 20.** For x, $40 \ge 20$ ✓; for y, $25 \ge 20$ ✓.
3. **{C}: v(C) = 30.** For x, $50 \ge 30$ ✓; for y, $85 \ge 30$ ✓.

4. **{A, B}: v(AB) = 50.** For x, *30 + 40 = 70 ≥ 50* ✓; for y, *10 + 25 = 35 ≥ 50* ✗. If {A, B} leave together they get 50; they block y.
5. **{A, C}: v(AC) = 60.** For x, *30 + 50 = 80 ≥ 60* ✓; for y, *10 + 85 = 95 ≥ 60* ✓.
6. **{B, C}: v(BC) = 70.** For x, *40 + 50 = 90 ≥ 70* ✓; for y, *25 + 85 = 110 ≥ 70* ✓.
7. **{A, B, C}: total = 120.** For x, *30 + 40 + 50 = 120* ✓; for y, *10 + 25 + 85 = 120* ✓.

Conclusion: the Shapley imputation x is in the core. y is outside the core, because {A, B} blocks it. Since the game is convex, the Shapley value being in the core is also consistent with what Theorem 10.3 leads us to expect.

Coalition S	Constraint	x = (30, 40, 50)	y = (10, 25, 85)
{A}	Σ ≥ 10	30 ✓	10 ✓
{B}	Σ ≥ 20	40 ✓	25 ✓
{C}	Σ ≥ 30	50 ✓	85 ✓
{A, B}	Σ ≥ 50	70 ✓	35 ✗
{A, C}	Σ ≥ 60	80 ✓	95 ✓
{B, C}	Σ ≥ 70	90 ✓	110 ✓
{A, B, C}	total = 120	120 ✓	120 ✓

Table 10.18: The seven core constraints and the check of the two imputations.

Table 10.18 summarizes the check of the two imputations. A single ✗ is enough to throw an imputation out of the core. To exclude an imputation from the core it suffices to find one blocking coalition; to show that it is in the core, all the constraints must be checked.

A shortcut check. If the upper bounds of 10.4.9 ($x_A \le 50$, $x_B \le 60$, $x_C \le 70$) and the lower bounds (the stand-alone values) are known, an imputation that sums to 120 can be checked by comparing six numbers. For the proportional split *(20, 40, 60)*, for example: $10 \le 20 \le 50$, $20 \le 40 \le 60$, $30 \le 60 \le 70$. All hold; the proportional split is in the core as well. In the logistics game the core is wide, and three of the four proposals of Table 10.13 (equal, proportional, Shapley) are in the core. In this game the core does not select a single imputation; it only eliminates unacceptable ones.

10.4.12 A game with an empty core: the majority game

In the logistics game the core was a wide hexagon. Not every game has a stable split of this kind.

Example 10.11

Three agents will share a prize of 1 unit. If any two agents agree, they get the prize. The characteristic function is: $v(S) = 1$ if $|S| \ge 2$, and 0 otherwise. This game is called the **majority game**.

1. **Write the constraints.** The singleton constraints are $x_i \geq 0$. The pair constraints: $x_A + x_B \geq 1$, $x_A + x_C \geq 1$, $x_B + x_C \geq 1$. Efficiency: $x_A + x_B + x_C = 1$.
2. **Add the pair constraints.** Each player appears in two constraints. The sum is $2(x_A + x_B + x_C) \geq 3$, that is, $x_A + x_B + x_C \geq 3/2$.
3. **Contradiction.** Efficiency requires the total to be exactly 1. But $1 < 3/2$. No imputation can satisfy all three pair constraints at once: **the core is empty**.
4. **The Shapley value.** The three players are symmetric; by the symmetry axiom they receive equal amounts, and by efficiency the total is 1. $\varphi = (1/3, 1/3, 1/3)$. This split is fair, but each pair gets only $2/3$ and could get 1 by leaving: it is not stable.

What an empty core means can be seen from a sequence of bargains. Suppose A and B agree and share the prize as $(½, ½, 0)$. C, left out, offers A $(0.6, 0, 0.4)$: A gets 0.6 instead of 0.5, C gets 0.4 instead of 0; both gain. Now B, left out this time, offers C $(0, 0.3, 0.7)$: C gets 0.7 instead of 0.4, B gets 0.3 instead of 0. Now A is the one left out and can make B a better offer. In every split at least one pair gets at most $2/3$ in total (because the three pairs sum to 2), and that pair wants to exclude the third and share the 1. The bargaining never ends.

The lesson: a stable split does not exist in every game. In that case the rule has to be relaxed. Instead of "no group can gain anything by leaving", one says "no group can gain too much by leaving". This idea is expressed by the **ε-core**: no coalition can gain more than ε by leaving, that is, $\Sigma_{i \in S} x_i \geq v(S) - \varepsilon$ for every S. In the majority game the pair constraints become $x_i + x_j \geq 1 - \varepsilon$. Adding the three gives $2 \geq 3(1 - \varepsilon)$, that is, $\varepsilon \geq 1/3$. For $\varepsilon = 1/3$ the imputation $(1/3, 1/3, 1/3)$ satisfies every pair constraint with exact equality: $2/3 = 1 - 1/3$. The set of imputations that make ε smallest is called the **least core**. The solution that continues this idea step by step and picks a single imputation in every game is called the **nucleolus**; in the majority game the nucleolus is also $(1/3, 1/3, 1/3)$.

10.4.13 Weighted voting and power indices

An important application of coalitional games is voting. In a parliament the parties' seat counts, in a company the partners' shareholdings, in a network the nodes' voting rights can differ. Does the one with more weight have more power? The answer is "not always".

Definition 10.14

A **weighted voting game** is written $[q; w_1, \ldots, w_n]$. Here w_i is the voting weight of i, and q is the **quota** needed for a decision to pass. If the total weight of a coalition reaches the quota, the coalition wins and $v(S) = 1$; otherwise it loses and $v(S) = 0$.

In an arrival order, the player who, together with those before it, raises the total from below the quota to the quota or above is called the **pivotal player**. A member of a winning coalition whose departure makes the coalition lose is called a **critical (swing) member**.

A player's power in voting is measured by how often it decides outcomes. Two classical indices measure this.

The **Shapley–Shubik index** (Shapley and Shubik, 1954) is the Shapley value of the weighted voting game. In a voting game the marginal contribution can only be 0 or 1: it is 1 if the player is the one who pushes the total past the quota, and 0 otherwise. So the Shapley value of i is the fraction of orders in which i is pivotal.

The **Banzhaf index**, by contrast, looks not at orders but at coalitions. In each winning coalition one counts which members are critical. A player's Banzhaf index is the ratio of its number of critical memberships to the sum of all players' critical memberships.

Example 10.12

A parliament has 100 seats: party A has 50, party B has 49, and party C has 1. A simple majority, that is, 51 votes, is needed for a decision. The game is *[51; 50, 49, 1]*.

1. **Winning coalitions.** No party reaches 51 alone. The pairs: $A + B = 99$ ✓, $A + C = 51$ ✓, $B + C = 50$ ✗. All three together: 100 ✓. The winning coalitions are {A, B}, {A, C}, {A, B, C}.
2. **Shapley–Shubik: six orders.** A → B → C: total 50 with A, 99 with B; pivotal B. A → C → B: 50 with A, 51 with C; pivotal C. B → A → C: 49 with B, 99 with A; pivotal A. B → C → A: 49 with B, 50 with C, 100 with A; pivotal A. C → A → B: 1 with C, 51 with A; pivotal A. C → B → A: 1 with C, 50 with B, 100 with A; pivotal A.
3. **The Shapley–Shubik index.** A is pivotal in four orders, B in one, C in one: *(4/6, 1/6, 1/6) = (2/3, 1/6, 1/6)*.
4. **Banzhaf: critical members.** In {A, B}, if A leaves 49 remains, if B leaves 50 remains; both are critical. In {A, C}, if A leaves 1 remains, if C leaves 50 remains; both are critical. In {A, B, C}, if A leaves 50 remains (loses); if B leaves 51 remains (still wins); if C leaves 99 remains (still wins). Only A is critical.
5. **The Banzhaf index.** The numbers of critical memberships are A 3, B 1, C 1; total 5. The index is *(3/5, 1/5, 1/5)*.

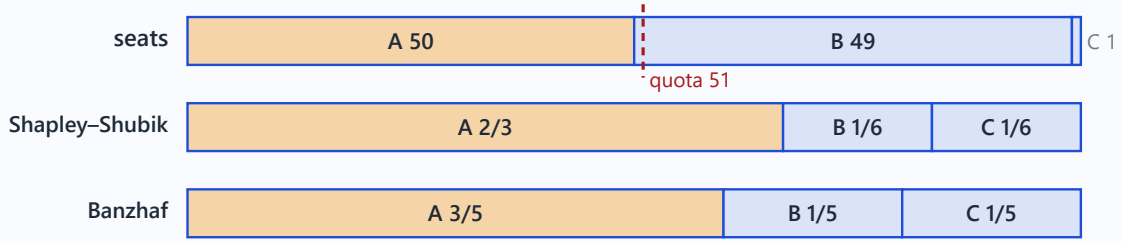


Figure 10.15: Seat shares and power indices in the parliament *[51; 50, 49, 1]*. B's 49 seats and C's 1 seat bring the same power: 1/6 in the Shapley–Shubik index and 1/5 in the Banzhaf index.

Figure 10.15 shows how striking the result is. B and C differ by 48 seats, yet the two parties are equally powerful. Why? The only way to win is a coalition that includes A, and for A a single seat is enough to pass the quota. B's extra 48 seats do not change the outcome of any coalition. Power is measured not by weight but by the number of situations in which the player can change the outcome. **Weight ≠ power.**

The two indices give different numbers because they count different things. The Shapley–Shubik index cares about the order in which coalitions are built and models the process by which the grand coalition forms. The Banzhaf index ignores order; it counts every winning coalition equally. Both find A the strongest and B and C equal. In terms of ranking, then, the two indices agree in this game.

A historical example: the 1958 Council of Ministers of the EEC. At the founding of the European Economic Community, the votes in the Council of Ministers were distributed as follows: France, Germany, and Italy 4 each; Belgium and the Netherlands 2 each; Luxembourg 1. A qualified majority required 12 votes: the game is *[12; 4, 4, 4, 2, 2, 1]*. Luxembourg's vote could not change the outcome of any coalition. The reason is an even–odd observation: all weights other than Luxembourg's are even, so the total of every coalition without Luxembourg is even. Luxembourg's 1 vote could carry a coalition from below the quota to the quota only if the total were exactly 11; but 11 is odd, and no coalition can have a total of 11. So Luxembourg is never pivotal and its power is 0. The Shapley–Shubik index of the other countries is 7/30 for each of the big three and 3/20 for each of Belgium and the Netherlands.

10.4.14 Code trace: power_index()

The Shapley–Shubik index is computed with simpler code than the code for the Shapley value: instead of looking up coalition values, the weights are added up and the player who passes the quota is recorded. The function takes three inputs: the player list `N`, the weight dictionary `w = {"A": 50, "B": 49, "C": 1}`, and the quota `q = 51`.

```
from itertools import permutations
def power_index(N, w, q):
  pivots = {i: 0 for i in N}
  orders = list(permutations(N))
  for order in orders:
    total = 0
    for i in order:
      total += w[i]
      if total >= q:
        pivots[i] += 1
        break
  return {i: pivots[i] / len(orders) for i in N}
```

The code works line by line as follows.

1. `pivots = {i: 0 for i in N}`: The number of orders in which each player is pivotal starts at 0.

2. `orders = list(permutations(N))` : All arrival orders (3! = 6) are generated.
3. `total = 0` : Each order starts with the empty coalition, that is, 0 votes.
4. `total += w[i]` : The weight of the arriving player is added to the total.
5. `if total >= q` : If the total has reached the quota for the first time, the arriving player is pivotal; `pivots[i]` is increased by one.
6. `break` : The pivot has been found in this order; the rest of the order does not matter. The inner loop is cut short and the next order begins. Each order has exactly one pivotal player.
7. `return {i: pivots[i] / len(orders) for i in N}` : The pivot counts are divided by the number of orders.

order	Running total	Pivot	pivots (A, B, C)
A → B → C	50 < 51 → 99 ≥ 51	B	(0, 1, 0)
A → C → B	50 < 51 → 51 ≥ 51	C	(0, 1, 1)
B → A → C	49 < 51 → 99 ≥ 51	A	(1, 1, 1)
B → C → A	49 → 50 < 51 → 100 ≥ 51	A	(2, 1, 1)
C → A → B	1 < 51 → 51 ≥ 51	A	(3, 1, 1)
C → B → A	1 → 50 < 51 → 100 ≥ 51	A	(4, 1, 1)

Table 10.19: Trace of `power_index(N, w, 51)`.

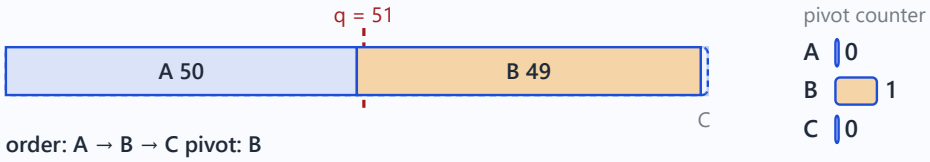


Figure 10.16: The first order, A → B → C. A's 50 votes fall short of the quota (dashed red line, 51); when B arrives the total becomes 99 and the quota is passed. The pivot is B (orange); the counter B = 1.



Figure 10.17: The end of the trace: the pivot counters are A 4, B 1, C 1.

Table 10.19, Figure 10.16, and Figure 10.17 show the trace. In the end `pivots = {A: 4, B: 1, C: 1}` and the function returns *(2/3, 1/6, 1/6)*. B's 49 seats bring no more power than C's 1 seat. The `break` line of the trace is important: in the order B → C → A, the total is 50 when B and C have arrived; when A arrives it reaches 100 and A is pivotal. Without `break`, the code would also count as pivotal the players who arrive after the quota has been passed, and the result would be wrong.

10.4.15 Common mistakes in coalitional games

Common mistake

Confusing the marginal contribution with the coalition value. In the order A → B → C, B's contribution is not $v(AB) = 50$ but $v(AB) - v(A) = 40$. The row total coming out to $v(N)$ catches this mistake: 10 + 50 + 70 = 130 ≠ 120.

Computing the Shapley value from a single order. A single order gives the last arrival a large share. The order A → B → C gives C 70; the Shapley value is 50. The average over all orders must be taken.

Checking only the singleton constraints for the core. $y = (10, 25, 85)$ satisfies all the singleton constraints, but {A, B} blocks it. Individual rationality is only one part of the core.

Assuming what is fair is stable. The Shapley value always exists, but it need not be in the core. In the majority game $(1/3, 1/3, 1/3)$ is fair, but the core is empty and every pair wants to leave.

Taking weight for power. In $[51; 50, 49, 1]$ the weight shares are $(0.50, 0.49, 0.01)$, whereas the Shapley–Shubik index is $(2/3, 1/6, 1/6)$. In the Luxembourg example, 1 vote brings no power at all.

Mixing up the two power indices. $(3/5, 1/5, 1/5)$ is the Banzhaf index (critical memberships); $(2/3, 1/6, 1/6)$ is the Shapley–Shubik index (pivotal orders). Whichever index the question asks for is the one to compute.

10.5 The public goods game

In coalitional games it was assumed that binding agreements could be made. In many multi-agent systems there is no such agreement: each agent decides on its own how much to contribute to a common resource, and nobody can force it to contribute. The public goods game is the many-player version of the Prisoner's Dilemma. The game shows with numbers why the common good does not arise by itself.

10.5.1 What is a public good?

Definition 10.15

A **public good** is a good with two properties:

- **Non-excludability:** nobody can be excluded from benefiting from the good. Those who do not contribute benefit too.
- **Non-rivalry:** one person's use does not reduce another's.

Open-source software is a good example. Whoever wrote the code and whoever did not can download and use the software (non-excludability). One person's use of the software does not reduce anyone else's (non-rivalry). Clean air and a shared dataset are also public goods. A loaf of bread, by contrast, is not a public good: someone who does not pay cannot get it, and bread eaten by one person cannot be eaten by another.

The problem with public goods arises from non-excludability. Since the good is open to everyone, each person may think: "Even if I do not contribute, others will, and I will benefit." If everyone thinks this way, nobody contributes and the good is never produced.

10.5.2 The rules of the game

Definition 10.16

In the **public goods game**, each of n players starts with e tokens. Player i puts k_i tokens, between 0 and e, into a common pool. The tokens in the pool are multiplied by a factor r and shared equally among the n players. Player i's payoff is

$$u_i = e - k_i + (r/n) \cdot \Sigma_j \, k_j$$

Here $\Sigma_j \, k_j$ is the total number of tokens put into the pool. r/n is the part of a token that returns to the contributor, and is called the **marginal per-capita return** (MPCR).

The payoff formula has two parts. $e - k_i$ is the tokens the player keeps in its pocket. $(r/n) \cdot \Sigma \, k_j$ is the share it receives from the pool. This share depends not on the player's own contribution but on everyone's total contribution. A player who does not contribute receives the same share from the pool; non-excludability is built into the formula.

The example of the chapter uses these numbers: $n = 4$ players, each with $e = 10$ tokens, multiplier $r = 1.6$. The marginal per-capita return is $r/n = 1.6/4 = 0.4$. A contribution of one token has two separate consequences.

Individual consequence

: For a player who contributes 1 token, 1 token leaves the pocket and 0.4 comes back from the pool. The net effect is $-1 + 0.4 = -0.6$.

Social consequence

: The 1 token becomes 1.6 in the pool and is distributed to the four players at 0.4 each. The group's total payoff rises by $-1 + 1.6 = +0.6$.

The individual and social consequences have opposite signs. Contributing hurts the player but benefits the group. This is the essence of the dilemma.

Example 10.13

Payoffs are computed in four scenarios ($n = 4$, $e = 10$, $r = 1.6$).

1. **Everyone gives 10.** The pool is $40 \cdot 1.6 = 64$ and the per-person share is $64/4 = 16$. Everyone gets $10 - 10 + 16 = 16$. The group's total is 64.
2. **Nobody gives.** The pool is 0. Everyone keeps the 10 in their pocket: 10 each, total 40.
3. **P1 free rides, the others give 10.** 30 tokens enter the pool: $30 \cdot 1.6 = 48$, 12 per person. P1 gets $10 - 0 + 12 = 22$, the others get $10 - 10 + 12 = 12$. Total $22 + 3 \cdot 12 = 58$.
4. **Only P1 gives 10.** 10 tokens enter the pool: $10 \cdot 1.6 = 16$, 4 per person. P1 gets $0 + 4 = 4$, the others get $10 + 4 = 14$. Total $4 + 3 \cdot 14 = 46$.

Scenario	Σk	Pool × 1.6	P1	Others (each)	Total
Everyone 10	40	64	16	16	64
Nobody	0	0	10	10	40
P1 free rides	30	48	22	12	58
Only P1	10	16	4	14	46

Table 10.20: Payoffs in four scenarios ($n = 4$, $e = 10$, $r = 1.6$).

Table 10.20 shows the dilemma plainly. For P1 the best case is when the others give and P1 does not (22). The worst case is when the others do not give and P1 does (4). For the group, the best case is that everyone gives (64) and the worst is that nobody gives (40). The Total column shows one more rule: the group's total is $n \cdot e + (r - 1) \cdot \Sigma k = 40 + 0.6 \cdot \Sigma k$. Every token that enters the pool raises the group's total by 0.6.

10.5.3 Why free riding?

If P1's payoff is written in terms of the other three players' total contribution K,

$$u_1 = 10 - k_1 + 0.4 \cdot (k_1 + K) = 10 + 0.4K - 0.6\, k_1$$

results. If P1 gives one more token, 1 token leaves its pocket and 0.4 comes back from the pool: net −0.6. This value does not depend on K. **Whatever the others do, every token P1 contributes costs P1 0.6.**

Theorem 10.4 (A dominant strategy in the public goods game)

If $r/n < 1$, then in the public goods game $k_i = 0$ is a dominant strategy for every player, and the unique Nash equilibrium of the game is that everyone gives 0. If $r > 1$, this equilibrium is not Pareto optimal: everyone giving e yields more for every player.

Proof. Player i's payoff, with the others' total contribution K, is written $u_i = e + (r/n)K - (1 - r/n)\, k_i$. Since $r/n < 1$, the coefficient of k_i, $-(1 - r/n)$, is negative. So whatever K is, reducing k_i raises the payoff, and the largest payoff is attained at $k_i = 0$. This shows that giving 0 is a dominant strategy.

Since every player's dominant strategy is 0, the profile in which everyone gives 0 is a Nash equilibrium. There is no other equilibrium: if in some profile a player contributes a positive amount, that player strictly increases its payoff by lowering its contribution to 0.

At the equilibrium every player gets e. If everyone gives e, the pool becomes $n \cdot e \cdot r$ and every player gets $r \cdot e$. Since $r > 1$, $r \cdot e > e$: everyone is better off. The equilibrium is not Pareto optimal. □

In the example, the only Nash equilibrium is that everyone gives 0, and the per-person payoff is 10. Yet if everyone gave 10, each would get 16. The equilibrium is not Pareto optimal. A player who benefits from the common good without contributing is called a **free rider**, and this behavior is called **free riding**.

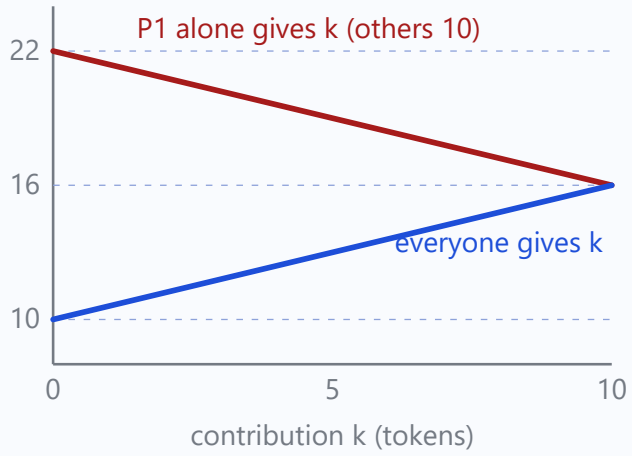


Figure 10.18: The effect of the contribution k on the payoff. Red line: while the others give in full (10), P1's payoff falls from 22 to 16 as P1 gives k. Blue line: if everyone gives the same k, the per-person payoff rises from 10 to 16.

Figure 10.18 draws the dilemma with two lines. The red line is P1's individual view: while the others give 10, $u_1 = 10 + 0.4 \cdot 30 - 0.6k = 22 - 0.6k$. Each token P1 gives lowers its payoff by 0.6; the best is not to give at all. The blue line is the group's view: if everyone gives k, $u = 10 - k + 0.4 \cdot 4k = 10 + 0.6k$. Each token that everyone gives raises the per-person payoff by 0.6; the best is for everyone to give in full. The slopes of the two lines have opposite signs: the individual incentive (−0.6) and social welfare (+0.6) point in opposite directions. This is the n-person version of the Prisoner's Dilemma.

10.5.4 When is there a dilemma?

The existence of the dilemma depends on two conditions. A contribution of one token returns r/n to the contributor and a total of r to the group.

- **Individual condition:** $r/n < 1$, that is, $r < n$. Contributing is individually harmful; not contributing is dominant.
- **Social condition:** $r > 1$, that is, $r/n > 1/n$. Each token raises the group's total by $r - 1 > 0$; contributing is socially beneficial.

The two conditions together, that is, $1 < r < n$ (equivalently $1/n < r/n < 1$), form the dilemma. In the two cases outside the condition there is no dilemma.

Example 10.14

Keep $n = 4$ and $e = 10$ fixed and vary r.

1. **r = 0.8 (r < 1).** A contribution of one token returns 0.2 to the contributor (net −0.8) and changes the group's total by $0.8 - 1 = -0.2$. Contributing harms both the individual and the group. Everyone giving 0 is both the equilibrium and the best for the group; there is no dilemma.
2. **r = 1.6 (1 < r < 4).** The main example: individual −0.6, social +0.6. There is a dilemma.
3. **r = 4 (r = n).** A contribution of one token returns exactly 1 to the contributor: net 0. The player's payoff does not change whether it contributes or not. Full contribution is weakly dominant: it is never worse. Socially, each token raises the group's total by 3.
4. **r = 6 (r > n).** One token returns $6/4 = 1.5$ to its giver: net +0.5. Full contribution is strictly dominant. The individual and social incentives point the same way; the dilemma disappears.

Example 10.14 gives a result that matters for design. One way to remove the dilemma is to raise the part of the contribution that returns to the individual above 1. Making the multiplier larger than the group size is often unrealistic for this purpose. But a reward tied to contribution (for example, priority for the contributor) creates the same effect: the contributor's personal return rises.

10.5.5 Counterparts in multi-agent systems

The public goods game is not an abstract laboratory game; it has direct counterparts in multi-agent systems.

File-sharing networks
: In a peer-to-peer (P2P) network, each peer downloads files and uploads files to others. Uploading uses bandwidth but benefits the network as a whole. A peer that only downloads without uploading is a free rider. If everyone behaves this way, no files remain in the network to be downloaded.

Federated learning
: Many clients (phones, hospitals, companies) train a common machine learning model together (federated learning). Each client sends the update it has computed on its own data. Sending an update is costly in terms of processing, energy, and data privacy. A client that uses the common model without sending updates is a free rider.

Shared computing and data pools
: A community builds a shared dataset or an open-source library. Many use it without contributing; the few who contribute carry the whole load.

The designers of these systems change the payoffs of the game in order to reduce free riding. Common ways are rewarding contribution (giving a peer that uploads a faster download right), excluding non-contributors (making the good partly excludable), or making the interaction repeated and keeping a reputation record. The common idea behind all of them is to turn the individual incentive in the same direction as social welfare by raising the individual return to contribution. This is the application of the incentive design principle of 10.2.6 to public goods.

10.5.6 Common mistakes about public goods

Common mistake

Forgetting to multiply or to divide the pool. When everyone gives 10, the pool is $40 \cdot 1.6 = 64$ and it is divided by four: 16. Taking the whole pool (64) to be one player's payoff, or forgetting the multiplier and writing $40/4 = 10$, are frequent mistakes.

Thinking the free rider gets no share of the pool. A public good is non-excludable. The free rider P1 also gets as much from the pool as the others (12); on top of that it keeps its 10 and reaches 22.

Thinking "if contributing is socially beneficial, individuals will contribute". The social return (+0.6) and the individual return (−0.6) are different. A rational individual looks only at its own return; this is the source of the dilemma.

Writing the dilemma condition wrongly. The dilemma exists when $1 < r < n$. If $r < 1$, contributing harms the group as well; if $r \geq n$, full contribution is (weakly) dominant. In both cases the individual and social incentives point the same way.

10.6 Summary

- An **agent** is a system with autonomy, reactivity, pro-activeness, and social ability (Wooldridge and Jennings, 1995). A **multi-agent system** consists of several agents that perceive and act in the same environment; each agent's information is local. In game theory an agent is a rational player that maximizes its expected utility.
- The **architecture** (reactive, deliberative, BDI, learning, hybrid) describes the inside of the agent; game theory describes the interaction among agents.
- Systems can be **cooperative** (the problem is coordination), **competitive** (the problem is strategy), or **mixed-motive**. Three small games represent this spectrum: two equilibria in the coordination game, a single inefficient equilibrium in the Prisoner's Dilemma, and only the mixed equilibrium (½, ½) in Matching Pennies.
- Most real systems use **hybrid control**: agents decide autonomously, while a center sets the rules, prices, or tenders. A **convention** selects one equilibrium of a coordination game in advance; a common signal such as a traffic light yields a **correlated equilibrium**.
- The environment of learning agents is **non-stationary**; simultaneous best response can enter an endless cycle in the corridor game. Simple local rules produce **emergent behavior** (Schelling's segregation model).
- The **contract net**: announcement → bid → award. It is distributed and simple and needs $O(n)$ messages per task, but it is greedy and sequential: in the example 18 min against an optimum of 13 min (38% more). The task order changes the result; re-tendering (swaps) reaches the optimum in two steps in the example.
- **Trying all assignments** takes $n!$ work (3,628,800 at n = 10); the Hungarian algorithm is $O(n^3)$. If the winner is paid its own bid, the contractor inflates its cost; if the second-lowest bid is paid, an honest bid is best.
- The **characteristic function** $v(S)$ gives the value of every coalition. In a superadditive game the grand coalition makes the total value largest; whether it stays together depends on the split. The number of coalition structures is the Bell number (5 for 3 agents, 15 for 4 agents, 115,975 for 10 agents).
- The **Shapley value** = the average marginal contribution; it is the only rule that satisfies the four axioms (efficiency, symmetry, dummy player, additivity). In the logistics example $\varphi = (30, 40, 50)$. In the pairwise synergy model φ_i = stand-alone value + half of the synergies: $(9, 6, 6)$ in the sample round of the game, with an equal-split error of 4.

- The **core**: imputations that no coalition wants to leave. In the logistics example it is a hexagon and contains the Shapley value (the game is convex). {A, B} blocks $y = (10, 25, 85)$. In the majority game the core is empty; the least ε-core, with $\varepsilon = 1/3$, is $(1/3, 1/3, 1/3)$.
- **In weighted voting, power ≠ weight**: in $[51; 50, 49, 1]$ the Shapley–Shubik index is $(2/3, 1/6, 1/6)$ and the Banzhaf index is $(3/5, 1/5, 1/5)$. In the 1958 EEC Council, Luxembourg's power was 0.
- **In the public goods game**, if $r/n < 1$, not contributing is dominant: the equilibrium gives 10 per person, versus 16 if everyone contributes in full. The dilemma exists when $1 < r < n$. A **free rider** (a peer that only downloads, a client that sends no updates) gains individually while the system loses.

Concept	Description	Example
Coordination game	Several equally good equilibria; the problem is which one will be played	Corridor game; "everyone keeps right"
Contract net	Announcement, bid, award; greedy and sequential	18 min; optimum 13 min
Shapley value	Average marginal contribution over all orders	(30, 40, 50); in the game (9, 6, 6)
Core	Imputations that no coalition blocks	hexagon; (10, 25, 85) outside
Empty core	No stable split	majority game
Power index	Share of pivotal orders or critical memberships	[51; 50, 49, 1]: (2/3, 1/6, 1/6)
Public goods game	If $r/n < 1$, not giving is dominant	equilibrium 10, optimum 16

Table 10.21: The concepts of the chapter at a glance.

10.7 Key terms

agent multi-agent system (MAS) autonomy reactivity pro-activeness social ability deliberative agent belief–desire–intention (BDI) mixed-motive coordination game convention / social law correlated equilibrium (CE) non-stationary fictitious play emergent behavior task allocation resource allocation Contract Net Protocol manager / contractor task announcement / bid / award greedy algorithm Hungarian algorithm re-tendering combinatorial auction coalition grand coalition coalition formation coalition structure Bell number characteristic function transferable-utility (TU) game superadditive imputation marginal contribution Shapley value efficiency axiom symmetry axiom dummy (null) player additivity core blocking coalition convex game balanced game Bondareva–Shapley theorem majority game ε-core nucleolus weighted voting game quota pivotal player critical (swing) member power index Shapley–Shubik index Banzhaf index public good public goods game marginal per-capita return (MPCR) free rider free riding tragedy of the commons

10.8 Test Yourself

1. In a pure coordination game, (Right, Right) gives both players 2, (Left, Left) gives both 1, and different choices give 0. What are the pure Nash equilibria? In the mixed equilibrium, with what probability does each player choose Right, what is the expected payoff, and how does it compare with the payoffs of the pure equilibria?

   Answer

   The pure equilibria are (Right, Right) = (2, 2) and (Left, Left) = (1, 1); in both, a player who switches sides drops to 0. Mixed equilibrium: let Player 2 play Right with probability q. For Player 1, Right gives $2q$ and Left gives $1 - q$; indifference $2q = 1 - q \Rightarrow q = 1/3$. By symmetry $p = 1/3$. The expected payoff is $2 \cdot (1/3)(1/3) + 1 \cdot (2/3)(2/3) = 2/9 + 4/9 = 2/3$. This is below both pure equilibria (2 and 1): the players often make mismatched choices. A convention ("everyone keeps right") raises the payoff to 2.

2. A manager puts tasks G1, G2, G3 out to tender in this order. Costs (min): for G1, R1 4, R2 5, R3 6; for G2, R1 2, R2 3, R3 6; for G3, R1 7, R2 4, R3 9. Each robot takes at most one task. What are the contract net's assignment and total? What is the centralized optimum, and by what percentage is the contract net above it?

   Answer

   G1: bids 4, 5, 6 → R1 (4). G2: R2 3, R3 6 → R2 (3). G3: only R3 remains → 9. Total $4 + 3 + 9 = 16$ min. The six assignments: (R1, R2, R3) 16; (R1, R3, R2) $4 + 6 + 4 = 14$; (R2, R1, R3) $5 + 2 + 9 = 16$; (R2, R3, R1) $5 + 6 + 7 = 18$; (R3, R1, R2) $6 + 2 + 4 = 12$; (R3, R2, R1) $6 + 3 + 7 = 16$. The optimum is 12 min: G1→R3, G2→R1, G3→R2. The contract net is $(16 - 12)/12 \approx 33\%$ above it. The loss again comes from greediness: R1, which took G1, was the cheapest robot for G2 (2).

3. For one task, the robots' true costs are R1 5, R2 7, R3 9, and the lowest bid wins. (a) If the winner is paid its own bid, what is R1's gain with an honest bid, and what happens if R1 raises its bid to 6.5? (b) If the winner is paid the second-lowest bid, what does an honest R1 earn? (c) Under this rule, what happens if R2 bids 4 in order to get the job?

   Answer

   (a) With an honest bid R1 wins, receives 5, and its gain is $5 - 5 = 0$. If it bids 6.5 (the others being honest), it is still the lowest bid; it wins and gains $6.5 - 5 = 1.5$. Under this rule inflating pays. (b) R1 wins and receives the second-lowest bid, 7: gain $7 - 5 = 2$. Any bid below 7 gives the same result. (c) R2 wins, but the payment is the second-lowest bid, R1's 5; since its cost is 7, its gain is $5 - 7 = -2$. With an honest bid it loses and gets 0; honesty is better.

4. In a game with three agents, $v(A) = 0$, $v(B) = 0$, $v(C) = 10$, $v(AB) = 30$, $v(AC) = 40$, $v(BC) = 40$, $v(ABC) = 60$. What is the Shapley value computed over the six orders? Is the game superadditive? Is it convex?

   Answer

   Marginal contributions (A, B, C): A→B→C (0, 30, 30); A→C→B (0, 20, 40); B→A→C (30, 0, 30); B→C→A (20, 0, 40); C→A→B (30, 20, 10); C→B→A (20, 30, 10). Totals A 100, B 100, C 160; $\varphi = (100/6, 100/6, 160/6) = (50/3, 50/3, 80/3) \approx (16.67, 16.67, 26.67)$, total 60 ✓. It is superadditive: $30 \geq 0$, $40 \geq 10$, $40 \geq 10$, and every pair plus a singleton is at most $40 + 0 = 40$ or $30 + 10 = 40$, which does not exceed 60. It is not convex: A's contribution to {B} is 30, while its contribution to the larger {B, C} is $60 - 40 = 20$; the contribution fell as the coalition grew.

5. In the game of Question 4, is the Shapley value in the core? Are the imputations $(20, 20, 20)$ and $(30, 10, 20)$ in the core? Which coalition blocks the one that falls outside?

   Answer

   Upper bounds $x_i \leq v(N) - v(N \setminus \{i\})$: $x_A \leq 60 - 40 = 20$, $x_B \leq 20$, $x_C \leq 60 - 30 = 30$; lower bounds 0, 0, 10. The Shapley value $(16.67, 16.67, 26.67)$ is within all the bounds: it is in the core. Although the game is not convex, the Shapley value is in the core; convexity is a sufficient condition, not a necessary one. $(20, 20, 20)$: within the bounds (A and B exactly at the upper bound), so it is in the core. $(30, 10, 20)$: $x_A = 30 > 20$; {B, C} get only $10 + 20 = 30 < 40$ and would leave and share 40. {B, C} blocks.

6. The weighted voting game is $[7; 4, 3, 2, 1]$ (A 4, B 3, C 2, D 1). What are the winning coalitions, the Banzhaf index, and the Shapley–Shubik index? How do the powers of C and D compare?

   Answer

   The total weight is 10. Winning coalitions: AB (7), ABC (9), ABD (8), ACD (7), ABCD (10). The best coalition without A is BCD = 6 < 7, so A is in every winning coalition. Critical memberships: in AB, A and B; in ABC, A and B (if C leaves, 7 remains); in ABD, A and B; in ACD, A, C, and D; in ABCD, A. The counts are A 5, B 3, C 1, D 1, total 10: Banzhaf $(1/2, 3/10, 1/10, 1/10)$. Shapley–Shubik (24 orders): pivot counts A 14, B 6, C 2, D 2 ⇒ $(7/12, 1/4, 1/12, 1/12)$. For example, C is pivotal only if exactly A and D come before it (total 5); this happens in $2! \cdot 1! = 2$ orders. C's weight is twice D's, but their powers are equal in both indices.

7. In a public goods game, $n = 3$, $e = 20$, $r = 1.5$. What are the payoffs if everyone gives 20, if nobody gives, and if P1 free rides while the others give 20? What are the marginal per-capita return and

the individual and social net effects of one token? Is there a dilemma?

Answer

Everyone gives 20: the pool is $60 \cdot 1.5 = 90$, 30 per person; everyone gets 30. Nobody gives: everyone gets 20. P1 free rides: the pool is $40 \cdot 1.5 = 60$, 20 per person; P1 gets $20 + 20 = 40$, the others get $0 + 20 = 20$. MPCR $r/n = 0.5$: a contribution of one token has a net effect of $-1 + 0.5 = -0.5$ on the individual and $-1 + 1.5 = +0.5$ on the group. Since $1 < 1.5 < 3$, there is a dilemma: not giving is dominant, and the equilibrium is 20 per person, versus 30 if everyone gives.

8. Four agents will share a prize of 1 unit; if at least three agents agree, they get the prize: $v(S) = 1$ if $|S| \geq 3$, and 0 otherwise. Is the core empty? What is the Shapley value?

Answer

There are four triple constraints: the shares of every triple sum to at least 1. Each agent is in three of the four triples; adding the constraints gives $3(x_A + x_B + x_C + x_D) \geq 4$, that is, total $\geq 4/3$. But the total must be 1: the core is empty. The four agents are symmetric; by symmetry and efficiency $\varphi = (1/4, 1/4, 1/4, 1/4)$. Every triple gets only $3/4$ under this split and would like to exclude the fourth and share the 1.

10.9 Further reading

- M. Wooldridge, *An Introduction to MultiAgent Systems* (2nd edition, 2009): the standard textbook that presents the agent concept, agent architectures (reactive, BDI, hybrid), communication protocols such as the contract net, and coalition formation from a computer engineering viewpoint with little mathematics.
- G. Chalkiadakis, E. Elkind, and M. Wooldridge, *Computational Aspects of Cooperative Game Theory* (2011): a short, well-organized book that studies the characteristic function, the Shapley value, the core, the nucleolus, and weighted voting games from the viewpoint of computational complexity.
- A. R. Karlin and Y. Peres, *Game Theory, Alive* (2017), the chapter on coalitions and the Shapley value: gives the Shapley axioms and the uniqueness proof step by step with many examples.
- Y. Shoham and K. Leyton-Brown, *Multiagent Systems: Algorithmic, Game-Theoretic, and Logical Foundations* (2009), Chapter 12: presents the solution concepts of cooperative game theory (the core, the Shapley value, the nucleolus) and the Bondareva–Shapley theorem formally; the PDF of the book is published free of charge on the authors' website.

10.10 References

1. O. N. Bondareva, “Some applications of linear programming methods to the theory of cooperative games”, *Problemy Kibernetiki* 10, 1963.
2. G. Chalkiadakis, E. Elkind, M. Wooldridge, *Computational Aspects of Cooperative Game Theory*, Morgan & Claypool, 2011.
3. G. de Clippel, K. Rozen, “Fairness through the lens of cooperative game theory: An experimental approach”, *American Economic Journal: Microeconomics* 14(3), 2022.
4. FIPA, *Contract Net Interaction Protocol Specification* (SC00029H), 2002.
5. D. B. Gillies, “Solutions to general non-zero-sum games”, *Contributions to the Theory of Games IV*, Princeton, 1959.
6. A. R. Karlin, Y. Peres, *Game Theory, Alive*, American Mathematical Society, 2017.
7. N. Nisan, T. Roughgarden, É. Tardos, V. Vazirani (eds.), *Algorithmic Game Theory*, Cambridge University Press, 2007 (Chapter 15: cost sharing, the core, and the Shapley value).
8. C. W. Reynolds, “Flocks, herds and schools: A distributed behavioral model”, *SIGGRAPH '87*, 1987.
9. T. Sandholm, K. Larson, M. Andersson, O. Shehory, F. Tohmé, “Coalition structure generation with worst case guarantees”, *Artificial Intelligence* 111, 1999.
10. T. C. Schelling, “Dynamic models of segregation”, *Journal of Mathematical Sociology* 1(2), 1971.
11. L. S. Shapley, “A value for n-person games”, *Contributions to the Theory of Games II*, Princeton, 1953.
12. L. S. Shapley, “On balanced sets and cores”, *Naval Research Logistics Quarterly* 14(4), 1967.
13. L. S. Shapley, “Cores of convex games”, *International Journal of Game Theory* 1, 1971.
14. L. S. Shapley, M. Shubik, “A method for evaluating the distribution of power in a committee system”, *American Political Science Review* 48(3), 1954.
15. Y. Shoham, K. Leyton-Brown, *Multiagent Systems: Algorithmic, Game-Theoretic, and Logical Foundations*, Cambridge University Press, 2009 (Chapter 12).
16. R. G. Smith, “The contract net protocol: High-level communication and control in a distributed problem solver”, *IEEE Transactions on Computers* C-29(12), 1980.
17. M. Wooldridge, *An Introduction to MultiAgent Systems*, 2nd edition, Wiley, 2009.
18. M. Wooldridge, N. R. Jennings, “Intelligent agents: Theory and practice”, *Knowledge Engineering Review* 10(2), 1995.

PART IV · CHAPTER 11

Algorithmic Fairness and Ethics

Algorithms that approve loan applications, screen résumés, or compute a defendant's risk score make decisions that directly affect people's lives. These decisions can be "accurate" and still load errors onto some groups more than others, systematically. These notes build the ways of measuring fairness on a single numerical example and show why the measures cannot all be satisfied at once. The notes also cover the fair division of resources (cake cutting and the division of indivisible goods), ethical principles, auditing, and people's strategic response to an algorithm.

LEARNING OUTCOMES

Computing the criteria
Compute demographic parity, equalized odds, predictive parity, and calibration from the confusion matrices of two groups.

Showing the incompatibility
Show numerically why the criteria cannot be satisfied together when base rates differ.

Fair division
Apply the cut-and-choose method; distinguish proportionality from envy-freeness.

Ethics and incentives
Discuss how transparency, accountability, auditing, and strategic behavior affect design.

11.1 Introduction

Mechanism design and game theory are mostly concerned with efficiency, that is, with whether a mechanism enlarges total welfare. The question of these notes is different: how are the consequences of a decision or an allocation *distributed* among individuals and groups? This question arises in two separate areas. The first is the yes/no decisions made by machine learning models. The second is the division of a resource (time, land, an inheritance, course seats) among people. In both areas the word "fair" has no single meaning. So one must first define the different meanings separately, and then study the relations among them.

11.1.1 A warm-up game: the Fair Division Game

The chapter's interactive game, the Fair Division Game, rests on a simple rule. Two players pick indivisible items one at a time, taking turns. Each item's card shows the value each of the two players assigns to it. The fixed sample round of the game uses four items; in the other levels the values are drawn at random in each round.

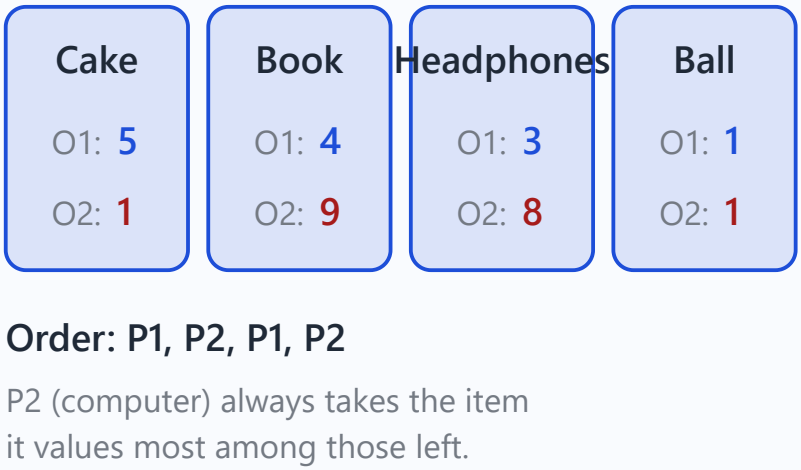


Figure 11.1: The sample round of the Fair Division Game. Each card shows the value that the first player (P1) and the second player (P2) assign to the item: Cake 5/1, Book 4/9, Headphones 3/8, Ball 1/1. The picking order is P1, P2, P1, P2; P2 (the computer) takes, each time, the remaining item that is most valuable in its own eyes.

The values in Figure 11.1 show that the two players value the items very differently. P1 likes the Cake most (5), and P2 likes the Book most (9). The game asks two questions:

- If the opponent always takes its own favorite item, is it best for P1 also always to take the most valuable item?
- Is there always an allocation in which everyone values their own share at least as much as they value the other's share?

The first question is a game-theoretic one: P1's strategy should be a best response to the opponent's behavior. The second is a fairness question: is an allocation in which "nobody envies anyone else's share" always possible? The answer to both questions is "no". The reasons are worked out in 11.4.8 and 11.4.9. The game shows two things from the start: fairness cannot be described with a single number, and even players who follow the rules can behave strategically.

11.1.2 Why do algorithmic decisions matter?

Algorithms now either make high-impact decisions directly or steer them with the recommendations they produce. Loan approval, hiring, medical diagnosis, social media recommendations, and criminal justice are the main such areas. An error in these decisions can cost a person their home, their job, or their freedom.

Violations of fairness and ethics cause two kinds of harm. The first is direct harm: an applicant rejected unjustly cannot get a loan, and a defendant unjustly labeled "high risk" may stay in detention longer. The second is indirect harm: as such incidents become known, public trust in institutions and technology erodes. Once trust is lost, even useful applications are not accepted.

From the viewpoint of game theory, a decision rule is a **mechanism**. The rule turns inputs (the applicant's data) into outputs (approval or rejection). Around this mechanism there are stakeholders with conflicting interests: the applicant wants a loan, the institution wants to avoid loans that are not repaid, and society wants opportunities to be distributed equitably among groups. As in mechanism design, which objective to pursue is a design decision here too.

Three examples show how the problem appears in real life:

- **Amazon's hiring tool (2018).** The company's screening model, trained on past résumés, penalized applications that contained the word "women's" (for example "captain of the women's chess club"). The model had learned this pattern from data in which mostly men had been hired in the past. The tool was withdrawn (Dastin, 2018).
- **COMPAS (2016).** A recidivism risk score computed for defendants in the United States. Journalists showed that the score produced different error rates for Black and white defendants; the company argued that the score conveyed the same risk in both groups (Angwin et al., 2016). This debate is the best-known example of how fairness criteria conflict when base rates differ. It is worked out with numbers in 11.3.5.
- **Credit scoring.** Two questions stand out here. Does the same score indicate the same risk for everyone? Who cannot get credit at all? The first reflects the viewpoint of the institution, the second that of the applicant. The numerical example of the chapter is also a credit model.

11.1.3 Where do bias and discrimination come from?

For a model to produce unfair results, the person who wrote it need not have ill intent. **Bias** can seep in at many points in the process in which a model is developed and used. Four sources stand out.

Historical data
: Past inequalities are reflected in the data labels. The model treats these labels as the "right answer" and learns to reproduce them. The gender bias in the Amazon example is a case in point: the label "was hired in the past" recorded not merit but the hiring habits of the past.

Underrepresentation
: If a group is thinly represented in the data, the model makes more errors on that group. The model tries to minimize the total error in training; the errors on a small number of examples

contribute little to this total. In commercial face recognition systems, the highest error rate has been measured for darker-skinned women (Buolamwini and Gebru, 2018; the "Gender Shades" study).

Measurement and proxy variables
: Instead of the target one really wants to measure, an easily measured **proxy variable** may be used. For example, if "health spending" is measured instead of "health need", a group with less access to health care looks healthier because it spends less. The model systematically underestimates this group's need.

Design and feedback
: A wrongly chosen optimization objective, proxies for the protected attribute (for example the postal code), and the model's decisions changing the data that will be collected in the future belong to this group. The last case is a **feedback loop**; the code trace in 11.5.6 shows such a loop step by step.

An important lesson comes out of these sources. A **protected attribute**, that is, an attribute such as gender or ethnic origin on which discrimination is prohibited, does not solve the problem even if it is removed from the model. The other variables carry it indirectly. If neighborhoods in a city are segregated by ethnic origin, the postal code is a good proxy for ethnic origin. The model can reproduce the same group difference by using the postal code. Therefore the way to secure fairness is not to delete a column from the input but *to measure outcomes by group*.

Common mistake

- **"If the model does not see gender, it is fair."** This approach is called "fairness through unawareness", and it often fails because of proxy variables. Moreover, once the protected attribute is deleted, measuring outcomes by group also becomes harder.
- **"The data reflect reality; the model only learns it."** Data are a record of past decisions and of the ways of measuring. If the label is a proxy, or was collected through past decisions, then what we take to be "reality" is already distorted.
- **"Bias comes only from the training data."** The choice of the target, the setting of the threshold, and the context in which the model is used are also sources of bias.

11.2 Fairness criteria

To discuss whether a model is "fair", one must first agree on what is to be measured. All four basic criteria are derived from the confusion matrix, which summarizes the results of a model that makes binary (yes/no) decisions. The same two groups and the same model are used in all the computations. In this way the differences among the criteria come not from changing numbers but from changing the question that is asked.

11.2.1 The confusion matrix

Consider a credit model. For each applicant there are two things. The first is the actual outcome: does the person really repay the loan? This outcome is denoted Y; $Y = 1$ means "repays" and $Y = 0$ means "does not repay". The second is the model's decision: does the model approve the loan? This decision is denoted $\hat{Y}$ ("Y hat"); $\hat{Y} = 1$ means "approve" and $\hat{Y} = 0$ means "reject". The hat is a reminder that this is a prediction, not a true value.

The two binary variables have four combinations. Each applicant falls into one of these four boxes. The table that shows the number of people in the boxes is called the confusion matrix.

Definition 11.1

The **confusion matrix** is a 2 × 2 table showing the number of people in the four combinations of the actual outcome Y and the model's decision $\hat{Y}$:

- **TP** (true positive): repays and was approved ($Y = 1, \hat{Y} = 1$);
- **FN** (false negative): would repay but was rejected ($Y = 1, \hat{Y} = 0$);
- **FP** (false positive): does not repay but was approved ($Y = 0, \hat{Y} = 1$);
- **TN** (true negative): does not repay and was rejected ($Y = 0, \hat{Y} = 0$).

The total number of applicants is $n = TP + FP + FN + TN$.

The naming has a logic. The second letter (P or N) says what the model said: P is "positive", that is, approve; N is "negative", that is, reject. The first letter (T or F) says whether this statement was true or false. FN is "false negative": the model said "no" and was wrong, because the person would in fact have repaid.

Five rates are derived from the matrix. Each rate looks at a particular row or column of the matrix and answers a different question.

Rate	Definition	Question	Where in the matrix?
Selection rate	$(TP + FP) / n$	What fraction is approved?	approve column, whole group
TPR (true positive rate)	$TP / (TP + FN)$	What fraction of those who would repay were approved?	"repays" row
FPR (false positive rate)	$FP / (FP + TN)$	What fraction of those who would not repay were approved?	"does not repay" row
PPV (precision)	$TP / (TP + FP)$	What fraction of those approved repay?	approve column
Base rate p	$(TP + FN) / n$	Share of those who repay in the group	"repays" row, whole group

Table 11.1: Rates derived from the confusion matrix (credit example).

These rates can also be written as conditional probabilities. The conditional probability $P(A \mid B)$ means "the probability of A given that B is known" and is the share of A among only those who satisfy B. TPR is the share of those approved among those who repay: $TPR = P(\hat{Y} = 1 \mid Y = 1)$. FPR is the share of those approved among those who do not repay: $FPR = P(\hat{Y} = 1 \mid Y = 0)$. PPV reverses the direction: it is the share of those who repay among those approved: $PPV = P(Y = 1 \mid \hat{Y} = 1)$. To avoid confusing TPR with PPV, the rule "the condition to the right of the bar says which crowd we are counting in" helps. In TPR the crowd is those who repay; in PPV it is those who are approved. The denominators of these two rates differ, and they generally give different numbers.

TPR is also called **sensitivity** in medicine and statistics. Another name for PPV is **precision**. The base rate is independent of the model: it does not look at what the model decides, and only measures the frequency of the actual outcome in the group. Even if the model changes, the base rate does not. One of the main results of the chapter is that when base rates differ, the criteria cannot be satisfied together (11.3).

Every group fairness criterion asks for one of these rates to be equal across groups. Which rate is to be equalized is not a technical detail but a *choice of values*: it is an answer to which error matters more, and to whom.

11.2.2 Example: two groups, one model

The example used throughout the chapter is this. There are 100 applicants in each of two groups, A and B. The model gives each applicant one of two scores: 0.8 ("low risk") or 0.2 ("high risk"). The model decides with a threshold of 0.5: those with a score above 0.5 are approved, and the others are rejected. The table below shows how many people in each score class actually repay.

group	score	people	repay	default	decision (threshold 0.5)
A	0.80	50	40	10	approve
A	0.20	50	10	40	reject
B	0.80	20	16	4	approve
B	0.20	80	16	64	reject

Table 11.2: Sample data. 100 applicants in each group; the model gives two scores.

The first observation from the table is that the two groups really are different. In group A, those who repay number 40 + 10 = 50; the base rate is $p_A = 50/100 = 0.50$. In group B, those who repay number 16 + 16 = 32; the base rate is $p_B = 0.32$. The second observation is that the model gives a high score less often to group B: in A, 50 people get 0.8, while in B only 20 do.

Example 11.1

The confusion matrices of the two groups are built from Table 11.2.

1. **Those approved.** Those with a score of 0.8 are approved. In A, 40 of them repay (TP = 40) and 10 do not (FP = 10). In B, 16 repay (TP = 16) and 4 do not (FP = 4).
2. **Those rejected.** Those with a score of 0.2 are rejected. In A, 10 of them are people who would in fact repay (FN = 10), and 40 are correctly rejected (TN = 40). In B there are 16 FN and 64 TN.
3. **Check.** The four cells of each matrix must sum to 100: in A, 40 + 10 + 10 + 40 = 100; in B, 16 + 4 + 16 + 64 = 100 ✓.
4. **Accuracy.** The people for whom the model decides correctly are in the TP and TN boxes. **Accuracy** is *(TP + TN)/n*. In A it is *(40 + 40)/100 = 0.80*, in B *(16 + 64)/100 = 0.80*.

The model is 80% accurate in both groups. At first glance this gives the impression that the model treats the two groups equally. But that impression is incomplete; the criteria bring out what is missing, in numbers.



Figure 11.2: The confusion matrices of the two groups. Rows show the actual outcome (repays $Y = 1$, defaults $Y = 0$), columns the model's decision (approve $\hat{Y} = 1$, reject $\hat{Y} = 0$). A: TP = 40, FN = 10, FP = 10, TN = 40. B: TP = 16, FN = 16, FP = 4, TN = 64.

Figure 11.2 shows the two matrices side by side. The *rows* of the matrix are separated by the truth and the *columns* by the decision. This layout makes it easier to see which cells the later criteria look at: rates computed along a row (TPR, FPR) ask "given the actual outcome, what did the model do?", while those computed along a column (PPV) ask "given the decision, what did the truth turn out to be?".

11.2.3 Demographic parity

The simplest fairness idea is that the two groups be approved at equal rates. This idea does not look at how accurate the model is; it looks only at the distribution of outcomes.

Definition 11.2

Demographic parity means that the selection rates of the two groups are equal:

$$P(\hat{Y} = 1 \mid A) = P(\hat{Y} = 1 \mid B).$$

Here $P(\hat{Y} = 1 \mid A) = (TP_A + FP_A) / n_A$ is the **selection rate** of group A.

Example 11.2

1. **The approve column.** The selection rate looks only at the approve column. In A the number approved is 40 + 10 = 50, in B it is 16 + 4 = 20.
2. **Selection rates.** A: 50/100 = 0.50. B: 20/100 = 0.20. The difference is 0.50 − 0.20 = 0.30. Demographic parity does not hold.
3. **The four-fifths rule.** The ratio of the smaller selection rate to the larger is 0.20 / 0.50 = 0.40. Since this number is below 0.8, the rule defined below is violated as well.
4. **The groups differ.** In reality those who repay number 50 in A and 32 in B. Raising the selection rate in B to 0.50 means raising the number approved from 20 to 50. Since only 32 people in B repay, at least 50 − 32 = 18 of these new approvals go to people who will not repay. Conversely, lowering A to 0.20 means rejecting at least 30 of the 50 people who would repay.

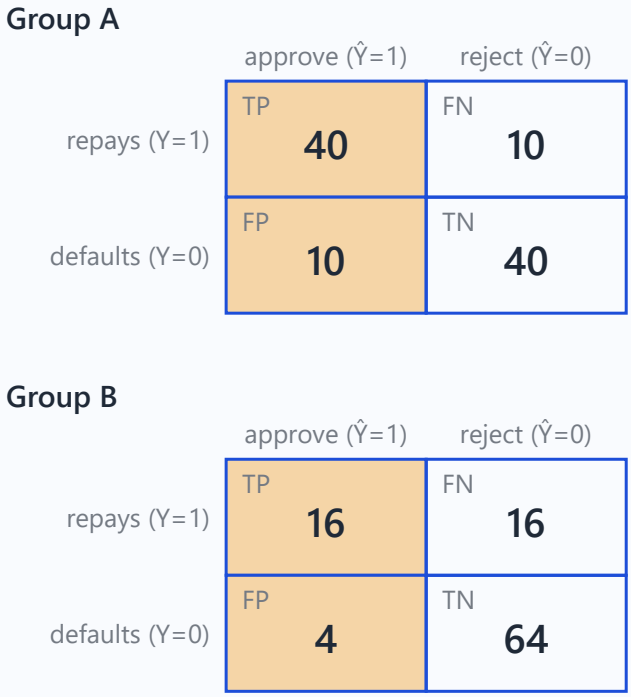


Figure 11.3: The cells that demographic parity looks at: the approve column (TP + FP). In A, 50 people are approved; in B, 20; the selection rates are 0.50 and 0.20.

In US employment law a practical version of this criterion is used. In hiring, if the selection rate of a group is less than four-fifths (80%) of the selection rate of the group with the highest rate, this is taken as first evidence of disparate impact.

Definition 11.3

The **four-fifths rule** requires that the ratio of the smaller selection rate to the larger be at least 0.8:

$$\min(s_A, s_B) / \max(s_A, s_B) \geq 0.8.$$

The rule looks at a *ratio*, not a *difference*. This distinction matters. If the selection rates are 0.50 and 0.40, the difference is 0.10 and the ratio is 0.8; the rule is met at the limit. If the selection rates are 0.10 and 0.02, the difference is only 0.08 but the ratio is 0.2; the rule is clearly violated. At small rates a difference looks misleadingly small.

The attractive side of demographic parity is that it does not require knowing the actual outcome Y. If the labels themselves carry historical bias (11.1.3), a criterion that does not rely on labels is advantageous. Its weak side shows in the last step above: if the base rates really differ, forcing this criterion means either giving credit to people who will not repay or rejecting people who would repay. Both are a kind of error, and the cost of these errors is also placed on some group.

11.2.4 Equalized odds and equal opportunity

The second idea takes the actual outcome into account. The question is: does the chance of being approved of two people in the same actual situation change with their group? If of two people who will repay one is in A and the other in B, their chances of approval should be equal. The same should hold for people who will not repay.

Definition 11.4

Equalized odds (Hardt et al., 2016) means that the error rates, conditional on the actual outcome, are equal across groups:

$$TPR_A = TPR_B \text{ and } FPR_A = FPR_B.$$

If only TPR equality is required, the criterion is called **equal opportunity**.

The name of equal opportunity describes its content: the opportunity of everyone who "deserves" approval, that is, who will repay the loan, to get approved should be independent of their group. Equalized odds adds a second condition: the chance that those who do not deserve it are approved by mistake should also be independent of group.

Example 11.3

1. **TPR (the "repays" row).** A: $40/(40 + 10) = 40/50 = 0.80$. B: $16/(16 + 16) = 16/32 = 0.50$. In B, half of those who will repay the loan are rejected; in A this share is one fifth.
2. **FPR (the "does not repay" row).** A: $10/(10 + 40) = 10/50 = 0.20$. B: $4/(4 + 64) = 4/68 \approx 0.059$.
3. **Comparison.** The TPR gap is 0.80 − 0.50 = 0.30 and the FPR gap is 0.20 − 0.059 = 0.141. Since both rates differ, equalized odds does not hold. Equal opportunity, which looks only at TPR, does not hold either.

The result paints an interesting picture: group B is treated differently from A in both directions. Members of B who will repay are rejected more often (low TPR); members of B who will not repay are approved less often (low FPR). The model is, overall, "more cautious" toward B.



Figure 11.4: The cells that equalized odds looks at. Top row: the "repays" row for TPR ($40/50 = 0.80$ in A, $16/32 = 0.50$ in B). Bottom row: the "does not repay" row for FPR ($10/50 = 0.20$ in A, $4/68 \approx 0.059$ in B).

Figure 11.4 shows that this criterion works by rows. The viewpoint of this criterion is that of the applicant. An applicant who would repay asks: "As someone who would repay, does my chance of approval depend on my group?" The chance of an applicant who would repay is 80% in A and 50% in B. This difference is explained not by the applicant's own behavior but only by the group.

The complement of TPR is also used often. The **false negative rate** (FNR) is the share of those rejected among those who would repay: $FNR = FN/(TP + FN) = 1 - TPR$. In the example FNR is 0.20 in A and 0.50 in B. Equality of TPR and equality of FNR are the same condition; the FNR form is used in the COMPAS debate (11.3.5).

11.2.5 Predictive parity

The third idea moves the viewpoint to the institution's side. The institution uses the model's "approve" decision as a piece of information. The meaning of this information should be the same in the two groups: an approved member of A and an approved member of B should repay the loan with the same probability.

Definition 11.5

Predictive parity means that the rate at which those approved actually repay, that is, the PPV, is equal across groups:

$$PPV_A = PPV_B, \text{ where } PPV = TP/(TP + FP) = P(Y = 1 \mid \hat{Y} = 1).$$

The counterpart on the reject side is the **negative predictive value** (NPV): the share of those rejected who really do not repay, $NPV = TN/(TN + FN)$.

Example 11.4

1. **PPV (the approve column).** A: $40/(40 + 10) = 0.80$. B: $16/(16 + 4) = 16/20 = 0.80$. In both groups 80% of those approved repay; predictive parity holds.
2. **NPV (the reject column).** A: $40/(40 + 10) = 40/50 = 0.80$. B: $64/(64 + 16) = 64/80 = 0.80$. Of those rejected, too, 80% in both groups really do not repay.

The labels "approve" and "reject" carry the same meaning in the two groups. For the institution the model is equally reliable in both groups.

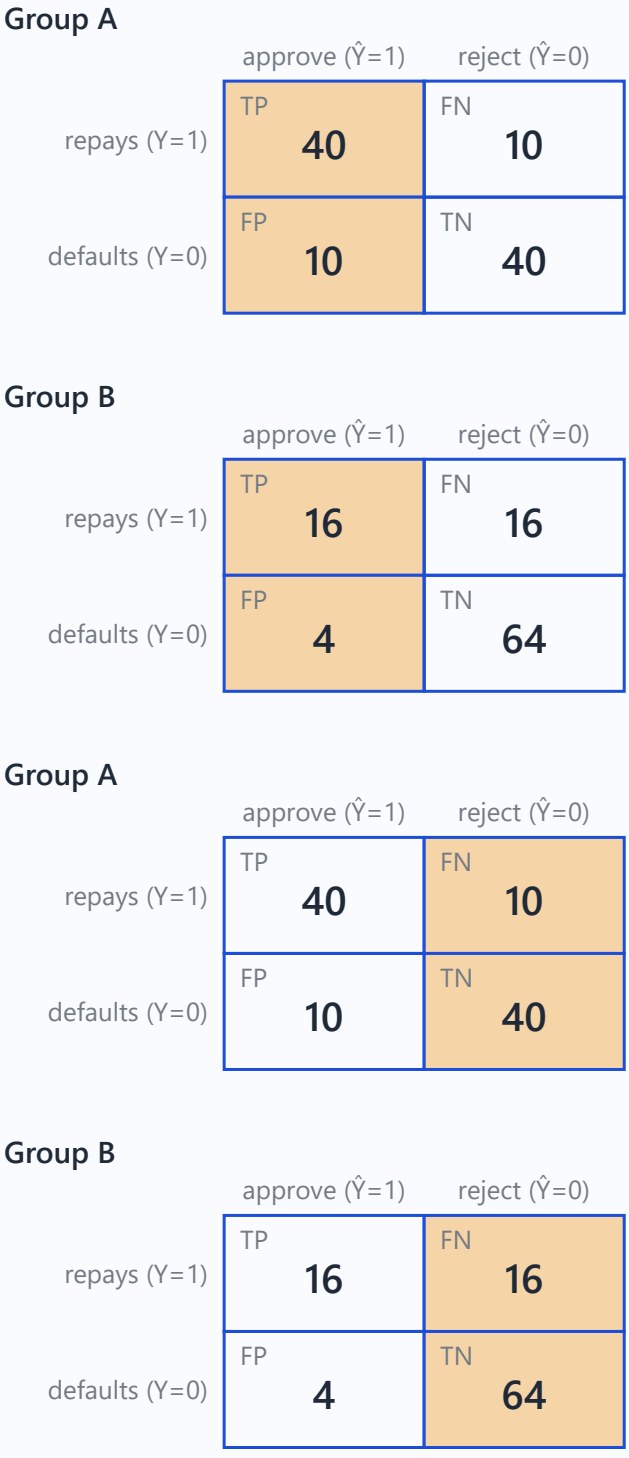


Figure 11.5: The cells that predictive parity looks at. Top row: the approve column for PPV (40/50 in A, 16/20 in B; both 0.80). Bottom row: the reject column for NPV (40/50 in A, 64/80 in B; both 0.80).

The same model, the same numbers: by equalized odds, group B is disadvantaged; by predictive parity, the model behaves the same toward both groups. There is no contradiction; the two criteria look at different directions of the matrix. Figure 11.4 compares rows and Figure 11.5 compares columns. Which direction matters depends on whose viewpoint is given priority.

11.2.6 Calibration

The criteria so far looked at the model's approve/reject *decision*. But the model first produces a *score*; the decision is made by comparing this score with a threshold. The score itself also makes a claim: "this person's probability of repaying is 0.8." Calibration asks whether this claim is true.

Definition 11.6

Calibration (calibration within groups): the actual repayment rate of people with score s is s in every group:

$$P(Y = 1 \mid \text{score} = s, \text{group} = g) = s, \text{ for every } s \text{ and every } g.$$

Calibration means that the score conveys the same risk whatever the group. Consider a weather app: if of the days on which it says "30% chance of rain" it actually rains on about 30%, the app is calibrated. That calibration is within groups means that this check is made separately for each group. A model can be calibrated overall and still systematically inflate the scores in one group.

Example 11.5

For each of the four rows of Table 11.2, the realized rate is computed and compared with the score.

1. **Group A, score 0.8:** 40 of 50 repay ⇒ 40/50 = 0.80. Same as the score.
2. **Group A, score 0.2:** 10 of 50 repay ⇒ 10/50 = 0.20. Same as the score.
3. **Group B, score 0.8:** 16 of 20 repay ⇒ 16/20 = 0.80. Same as the score.
4. **Group B, score 0.2:** 16 of 80 repay ⇒ 16/80 = 0.20. Same as the score.

In all four rows the score equals the realized rate. The model is calibrated in both groups: the score is an honest probability.

group	score s	people	repay	realized rate	equals s?
A	0.80	50	40	40/50 = 0.80	yes
A	0.20	50	10	10/50 = 0.20	yes
B	0.80	20	16	16/20 = 0.80	yes
B	0.20	80	16	16/80 = 0.20	yes

Table 11.3: The calibration check. Realized rate = repay / people.

Calibration and predictive parity are closely related but not the same. Predictive parity looks only at the "approve" column, that is, at everyone above the threshold together. Calibration looks at each score level separately. In this example there is a single score above the threshold (0.8), so calibration directly gives PPV equality: all those approved have the score 0.8, and in a calibrated model 80% of them repay. In a model with more score levels, the mix of scores among those approved can differ in the two groups; then even a calibrated model can give different PPVs.

A calibrated model has an interesting property: the average score in a group equals the base rate of that group. In A the average score is $(50 \cdot 0.8 + 50 \cdot 0.2)/100 = 0.50 = p_A$, in B it is $(20 \cdot 0.8 + 80 \cdot 0.2)/100 = 0.32 = p_B$. For this reason, if the base rates differ, a calibrated model *must* give the two groups different scores on average. This observation is the core of Kleinberg's theorem in 11.3.3.

11.2.7 Code trace: metrics()

All four group criteria are computed from the four numbers of the confusion matrix. The short Python function below does this computation and prints the results of the two groups side by side.

```
def metrics(TP, FP, FN, TN):
  n = TP + FP + FN + TN
  sel = (TP + FP) / n     # demographic parity
  tpr = TP / (TP + FN)    # equalized odds
  fpr = FP / (FP + TN)
  ppv = TP / (TP + FP)    # predictive parity
  return sel, tpr, fpr, ppv

A = metrics(40, 10, 10, 40)
B = metrics(16, 4, 16, 64)
NAMES = ["sel", "TPR", "FPR", "PPV"]
for i, name in enumerate(NAMES):
  print(name, round(A[i], 3), round(B[i], 3))
```

The function takes four arguments: TP, FP, FN, and TN. In the first line the total number of people `n` is computed. The next four lines apply the definitions of Table 11.1 one to one. The comments say in which criterion each rate is used: `sel` is compared in demographic parity, `tpr` and `fpr` in equalized odds, and `ppv` in predictive parity. The function returns the four numbers as a Python tuple. A tuple is an immutable, ordered list of comma-separated values; `A[0]` gives the first value and `A[3]` the last.

The lower part of the program calls the function with the matrices of the two groups. The line `for i, name in enumerate(NAMES)` gives, for each name in the list, both its position (`i`) and the name itself (`name`). So `A[i]` and `B[i]` are the values of the same rate in the two groups.

Line	Group A	Group B
`n`	40 + 10 + 10 + 40 = 100	16 + 4 + 16 + 64 = 100
`sel`	(40 + 10)/100 = 0.5	(16 + 4)/100 = 0.2

Line	Group A	Group B
`tpr`	40/(40 + 10) = 0.8	16/(16 + 16) = 0.5
`fpr`	10/(10 + 40) = 0.2	4/(4 + 64) ≈ 0.0588
`ppv`	40/(40 + 10) = 0.8	16/(16 + 4) = 0.8
return	(0.5, 0.8, 0.2, 0.8)	(0.2, 0.5, 0.0588..., 0.8)

Table 11.4: Trace of `metrics()`. A = metrics(40, 10, 10, 40), B = metrics(16, 4, 16, 64).

The loop prints four lines: `sel 0.5 0.2`, `TPR 0.8 0.5`, `FPR 0.2 0.059`, `PPV 0.8 0.8`. Reading the output line by line, the status of the criteria is immediately visible. At `i = 0` the selection rates differ (0.5 ≠ 0.2): no demographic parity. At `i = 1` the TPRs differ: no equal opportunity. At `i = 2` the FPRs differ as well: no equalized odds. At `i = 3` the PPVs are equal: predictive parity holds.

Example 11.6

Let the matrix of a third group, C, be TP = 30, FP = 20, FN = 10, TN = 40. The function is called as `metrics(30, 20, 10, 40)`.

1. *n = 30 + 20 + 10 + 40 = 100*.
2. Selection rate *(30 + 20)/100 = 0.5*.
3. TPR = *30/(30 + 10) = 30/40 = 0.75*.
4. FPR = *20/(20 + 40) = 20/60 ≈ 0.333*.
5. PPV = *30/(30 + 20) = 30/50 = 0.6*.

Let us compare C with A: A's four values are (0.5, 0.8, 0.2, 0.8). Only the selection rates are equal (0.5 = 0.5); demographic parity holds. TPR (0.75 ≠ 0.8), FPR (0.333 ≠ 0.2), and PPV (0.6 ≠ 0.8) differ. So the two groups being approved at the same rate does not mean the approvals are handed out with the same accuracy.

11.2.8 All criteria, one model: the scorecard

All the computations can be gathered in one table. The table shows plainly that the same model can count as fair by one criterion and unfair by another.

Criterion	What must be equal	A	B	Satisfied?
Demographic parity	$P(\hat{Y} = 1)$	0.50	0.20	no
Equal opportunity	TPR = $P(\hat{Y} = 1 \mid Y = 1)$	0.80	0.50	no
Equalized odds	TPR and FPR = $P(\hat{Y} = 1 \mid Y = 0)$	0.80; 0.20	0.50; 0.059	no
Predictive parity	PPV = $P(Y = 1 \mid \hat{Y} = 1)$	0.80	0.80	yes

Criterion	What must be equal	A	B	Satisfied?
Calibration	$P(Y = 1 \mid score = s) = s$	0.8→0.8; 0.2→0.2	0.8→0.8; 0.2→0.2	yes

Table 11.5: All criteria, one model. The status of the example model under all the criteria.

The criteria in the table fall into two viewpoints:

- **The applicant's viewpoint:** "As someone who would repay, does my chance of approval depend on my group?" This question holds the actual outcome fixed and looks at the distribution of the decision ⇒ TPR and FPR (equal opportunity, equalized odds).
- **The institution's viewpoint:** "Does the repayment probability of a person I approve depend on their group?" This question holds the decision or the score fixed and looks at the distribution of the actual outcome ⇒ PPV and calibration.

Demographic parity falls outside these two viewpoints; it does not use the actual outcome at all and is concerned only with the numerical distribution of opportunities. The example model satisfies the institution's criteria and does not satisfy the applicant's. This is not a coincidence: it is unavoidable as long as the base rates differ (11.3).

11.2.9 Beyond group fairness: individual fairness

All of the criteria above are **group fairness** criteria: they compare the average rates of groups. While averages are being equalized, individuals can be treated arbitrarily. For example, additionally approving 30 randomly chosen people from group B raises B's selection rate to 0.50 and achieves demographic parity. But which 30 people get approved depends entirely on chance; of two very similar members of B, one is approved and the other is not. Dwork et al. (2012) approach this problem with a different principle: similar individuals should be treated similarly.

Definition 11.7

Individual fairness means that the model score satisfies the **Lipschitz condition**: for every two people x and y,

$$|f(x) - f(y)| \leq L \cdot d(x, y).$$

Here $f(x)$ is the model score of person x; $d(x, y)$ is the distance that measures how different the two people are from each other; L is the largest slope allowed.

The condition says that the difference in two people's scores must not exceed L times the difference between them. If $d(x, y)$ is small, that is, if the people are similar, their scores must also be close to each other. The name "Lipschitz" comes from a continuity condition in mathematics: functions whose

graph is nowhere steeper than L satisfy this condition. A function that makes a sudden jump violates the condition, because it gives very different values to two very close points on either side of the jump.

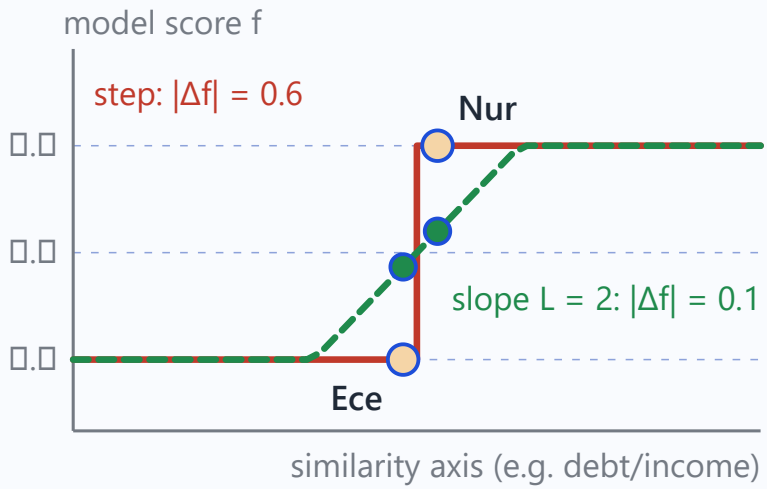


Figure 11.6: Individual fairness. The horizontal axis is a similarity measure (for example the debt-to-income ratio), the vertical axis is the model score. The red step model gives Ece and Nur, on either side of the threshold, 0.2 and 0.8: $|\Delta f| = 0.6$. The green dashed model rises with slope $L = 2$ and gives the same two people a difference of at most 0.1.

Example 11.7

In Figure 11.6 the distance between Ece and Nur is $d = 0.05$ and the allowed slope is $L = 2$.

1. **The largest allowed difference:** $L \cdot d = 2 \cdot 0.05 = 0.1$.
2. **The step model:** gives Ece 0.2 and Nur 0.8. The difference is $|0.8 - 0.2| = 0.6$. Since $0.6 > 0.1$, the condition is violated.
3. **The smooth model:** raises the score gradually so that the slope does not exceed 2. The difference in the two people's scores is at most $2 \cdot 0.05 = 0.1$; the condition holds.
4. **A question about a range:** let $L = 2$, $d(x, y) = 0.1$, and $f(x) = 0.8$. The condition gives $|f(y) - 0.8| \le 2 \cdot 0.1 = 0.2$; that is, $0.6 \le f(y) \le 1.0$. Since the upper bound is already the largest value a score can take, the allowed range is [0.6, 1].

Individual fairness is an attractive principle, but it has a difficulty: who determines the distance measure d? The question "who is similar?" is also a value judgment. Are two people with the same debt-to-income ratio similar, or does one person's irregular income make them different? Individual fairness does not remove the problem; it moves it to the choice of d. Moreover, the condition limits not the score itself but only the differences between scores. A model that gives everyone 0 also satisfies the condition. For this reason individual fairness is a constraint to be considered together with an accurate model.

11.2.10 Common mistakes in computing the criteria

Common mistake

- **Confusing TPR with PPV:** Both have TP in the numerator, but the denominators differ. In TPR the denominator is those who repay (TP + FN); in PPV it is those approved (TP + FP). In group B, TPR = 16/32 = 0.5 and PPV = 16/20 = 0.8.
- **Writing FN in the denominator of FPR:** The denominator of FPR is the total of those who do not repay: FP + TN. In B it is 4/(4 + 64) = 4/68, not 4/20.
- **Taking a difference in the four-fifths rule:** The rule asks for a ratio: smaller ÷ larger. The inverse ratio (larger ÷ smaller) is also wrong; the result 0.50/0.20 = 2.5 makes the rule look "satisfied".
- **Taking accuracy for fairness:** In the example the accuracy of both groups is 80%, yet the TPRs are 0.8 and 0.5. Equal accuracy does not show that the errors are of the same kind or fall on the same people.
- **Checking calibration in aggregate:** Calibration is checked separately within each group. A check made on the pooled data of the two groups can hide a systematic deviation in one of the groups.
- **Forgetting L in individual fairness:** The allowed difference is $L \cdot d$, not d.

11.3 Incompatibility of the criteria

The model in the previous section satisfied predictive parity and calibration but did not satisfy equalized odds. The first thought that comes to mind is that a better model could satisfy all of them at once. But this thought is generally wrong. Because the criteria are derived from the same four numbers, they are tied to one another. If the base rates differ, then apart from a few extreme cases no model can satisfy them all at once. Results of this kind are called **impossibility theorems**.

11.3.1 The Chouldechova identity

Chouldechova (2017) wrote the link among the criteria as a single equality. To derive the equality it suffices to express the confusion matrix of a group in terms of the base rate p, the TPR, and the FPR.

Theorem 11.1 (The Chouldechova identity)

In a group with base rate p, with $0 < p < 1$ and $PPV > 0$,

$$FPR = \frac{p}{1-p} \cdot \frac{1-PPV}{PPV} \cdot TPR.$$

Proof. **1. Writing the matrix cells with the rates.** Let the group have n people. By the definition of the base rate, those who repay number $p \cdot n$ and those who do not repay number $(1 - p) \cdot n$. Since TPR is the share of those who repay that are approved, $TP = p \cdot n \cdot TPR$. Since FPR is the share of those who do not repay that are approved, $FP = (1 - p) \cdot n \cdot FPR$.

2. Substituting into PPV. $PPV = TP / (TP + FP)$. Substituting the two expressions and canceling the common n in the numerator and denominator gives $PPV = p \cdot TPR / (p \cdot TPR + (1 - p) \cdot FPR)$.

3. Multiplying by the denominator. Multiply both sides by the denominator: $PPV \cdot [p \cdot TPR + (1 - p) \cdot FPR] = p \cdot TPR$. Expand the left side and move the term $PPV \cdot p \cdot TPR$ to the right: $(1 - p) \cdot FPR \cdot PPV = p \cdot TPR - p \cdot TPR \cdot PPV = p \cdot TPR \cdot (1 - PPV)$.

4. Isolating FPR. Divide both sides by $(1 - p) \cdot PPV$; this number is not zero because $p < 1$ and $PPV > 0$. The result is $FPR = p/(1 - p) \cdot (1 - PPV)/PPV \cdot TPR$. □

The identity ties the three rates (FPR, TPR, PPV) and the base rate together. If any three of them are known, the fourth can be computed. The numbers of the example model also follow this link.

Example 11.8

1. **Group A:** $p = 0.5$, $PPV = 0.8$, $TPR = 0.8$. The factors: $p/(1 - p) = 0.5/0.5 = 1$; $(1 - PPV)/PPV = 0.2/0.8 = 0.25$. Result: $FPR = 1 \cdot 0.25 \cdot 0.8 = 0.20$ ✓ (same as in the table).
2. **Group B:** $p = 0.32$, $PPV = 0.8$, $TPR = 0.5$. The factors: $0.32/0.68 = 8/17 \approx 0.471$; 0.25. Result: $FPR = (8/17) \cdot 0.25 \cdot 0.5 = 1/17 = 4/68 \approx 0.059$ ✓.
3. **Interpretation:** PPV is equal in the two groups (0.8), so the middle factor $(1 - PPV)/PPV$ is 0.25 in both. But the factor $p/(1 - p)$ is 1 in A and 0.471 in B.

The real question is this: when PPV is equal, can TPR and FPR be equal as well? Let PPV and TPR be equal in the two groups. Then on the right-hand side of the identity only the factor $p/(1 - p)$ varies from group to group. If p differs, this factor differs as well (because $p/(1 - p)$ increases strictly as p grows). As a result the FPRs come out different. For both the TPRs and the FPRs to be equal, the right-hand sides can be equalized only in extreme cases such as $TPR = 0$ or $PPV = 1$. $PPV = 1$ is the case in which everyone approved repays, that is, there is no false positive at all; $TPR = 0$ is the case in which nobody is given credit.

Theorem 11.2 (Incompatibility of predictive parity and equalized odds)

If the base rates of the two groups differ ($p_A \neq p_B$), predictive parity and equalized odds cannot hold together, except in extreme cases such as having no false positives at all (FPR = 0 and PPV = 1; perfect prediction is a special case of this), approving nobody, or none of those approved repaying (PPV = 0).

Proof. Let predictive parity and equalized odds hold together: $PPV_A = PPV_B = P$, $TPR_A = TPR_B = T$, $FPR_A = FPR_B$. Write the identity for the two groups and equate the FPRs: $p_A/(1 - p_A) \cdot (1 - P)/P \cdot T = p_B/(1 - p_B) \cdot (1 - P)/P \cdot T$. If $(1 - P)/P \cdot T$ is not zero, divide both sides by it, and what remains is $p_A/(1 - p_A) = p_B/(1 - p_B)$. Since this function is strictly increasing, we get $p_A = p_B$, which contradicts the assumption. So $(1 - P)/P \cdot T = 0$ must hold, that is, either $P = 1$ or $T = 0$. In both cases the identity gives FPR = 0. □

What the theorem says is not a model defect but an arithmetic fact. However much data is collected and however sophisticated an algorithm is used, as long as the base rates differ and prediction is imperfect, choosing one criterion means giving up another.

Example 11.9

Suppose that in a group $p = 0.2$, $PPV = 0.5$, $TPR = 0.6$ are given. What is FPR?

1. $p/(1 - p) = 0.2/0.8 = 0.25$.
2. $(1 - PPV)/PPV = 0.5/0.5 = 1$.
3. $FPR = 0.25 \cdot 1 \cdot 0.6 = 0.15$.

For a check, a group of 100 people can be set up: 20 repay, 80 do not. TP = 20 · 0.6 = 12, FP = 80 · 0.15 = 12. PPV = 12/24 = 0.5 ✓.

11.3.2 What if we force equalized odds?

To see concretely what the incompatibility means, let us go in the opposite direction. Suppose there is a hypothetical, better-discriminating model for group B that gives the same error rates as for A. In this model $TPR_B = 0.8$ and $FPR_B = 0.2$. The current model, because it gives only the scores 0.8 and 0.2, cannot reach these two error rates together by changing the threshold or by approving at random. We do not need to know how the new model reaches these rates; the confusion matrix can be computed from the error rates alone.

Example 11.10

1. **Target:** in B, TPR = 0.8 and FPR = 0.2 (the same as A). In B, 32 people repay and 68 do not.
2. **The rows of the matrix:** once the error rates are fixed, the top and bottom rows of the confusion matrix are determined. TP = 32 × 0.8 = 25.6; FP = 68 × 0.2 = 13.6. These are expected numbers, so they can be fractional.
3. **Selection rate:** approved = 25.6 + 13.6 = 39.2 people ⇒ selection rate ≈ 0.39. In B the approval rate rises from 0.20 to 0.39.
4. **PPV:** $PPV_B = 25.6/39.2 \approx 0.65$. In A, PPV was 0.80. The repayment rate of those approved falls in B.

Equalized odds was achieved, but predictive parity was broken: an "approve" in B now conveys a lower probability of repayment. Since the model gives only approve/reject, the criterion that breaks is PPV.

The result can also be found with the identity. Writing $0.2 = (8/17) \cdot (1 - PPV)/PPV \cdot 0.8$ for B gives $(1 - PPV)/PPV = 0.2 \cdot 17/(8 \cdot 0.8) = 0.53125$ and $PPV = 1/1.53125 \approx 0.653$. The two routes give the same result.

Mathematics does not answer the question of which criterion should be preserved. The answer depends on the context: which harm is heavier (rejecting someone who could repay, or putting someone who cannot repay into debt), and what the law requires. Giving credit to someone who cannot repay can also harm them; a loan that is not repaid ruins the person's credit history. For this reason the conclusion "more approvals are always fairer" does not follow either.

11.3.3 Calibration and balance: Kleinberg's theorem

The Chouldechova identity is for approve/reject decisions. Kleinberg, Mullainathan, and Raghavan (2017) studied the same problem at the level of scores. At the score level, the counterpart of equalized odds is **balance**.

Definition 11.8

Balance for the positive class: the average score of those who actually repay ($Y = 1$) is equal across groups. **Balance for the negative class**: the average score of those who actually do not repay ($Y = 0$) is equal across groups.

The meaning of balance for the positive class is similar to that of TPR equality: people in the same actual situation (those who repay) should receive the same score on average. The difference is that it looks at the score instead of the decision.

Example 11.11

1. **A, those who repay ($Y = 1$):** of the 50 who repay, 40 received 0.8 and 10 received 0.2. The average is $(40 \cdot 0.8 + 10 \cdot 0.2)/50 = (32 + 2)/50 = 0.68$.
2. **B, those who repay:** of the 32 who repay, 16 received 0.8 and 16 received 0.2. The average is $(16 \cdot 0.8 + 16 \cdot 0.2)/32 = (12.8 + 3.2)/32 = 0.50$. In our calibrated model, the members of B who repay get a lower score on average: no balance for the positive class.
3. **A, those who do not repay ($Y = 0$):** of the 50 who do not repay, 10 received 0.8 and 40 received 0.2. The average is $(10 \cdot 0.8 + 40 \cdot 0.2)/50 = (8 + 8)/50 = 0.32$.

4. **B, those who do not repay:** of the 68 who do not repay, 4 received 0.8 and 64 received 0.2. The average is $(4 \cdot 0.8 + 64 \cdot 0.2)/68 = (3.2 + 12.8)/68 = 16/68 \approx 0.235$. There is no balance for the negative class either.

Although calibration holds, both balances are broken.

Theorem 11.3 (Kleinberg, Mullainathan, Raghavan, 2017)

For a risk score, calibration within groups, balance for the positive class, and balance for the negative class can hold together in only two cases: if the prediction is perfect (every person gets a score of 0 or 1 and the score equals their actual outcome) or if the groups' base rates are equal.

The full proof of the theorem is long, but its main idea can be seen from the observation in 11.2.6. In a calibrated model the average score of a group equals its base rate. This average is the weighted average of the average of those who repay and the average of those who do not repay: $p = p \cdot$ *(average of those who repay)* $+ (1 - p) \cdot$ *(average of those who do not repay)*. If both balances hold, the two averages in parentheses are the same in the two groups; call these two numbers x and y. Then for each group $p = p \cdot x + (1 - p) \cdot y$, that is, $p \cdot (1 - x + y) = y$ must hold. If $1 - x + y$ is not zero, this equality ties p to a single value, and the base rates of the two groups come out equal. If the base rates differ, what remains is $1 - x + y = 0$; this is possible only with $y = 0$ and $x = 1$, that is, with perfect prediction (since the average scores lie between 0 and 1, $x - y = 1$ happens only this way). In our example the base rates differ (0.5 and 0.32) and the prediction is not perfect; the conflict of the criteria is unavoidable.

11.3.4 Is this model fair? The answer depends on the criterion

When the computations are put together, it turns out that the question "Is this model fair?" has no single answer. Two tables summarize the whole situation.

group	approve: repay / default	reject: repay / default	base rate
A	40 / 10	10 / 40	0.50
B	16 / 4	16 / 64	0.32

Table 11.6: The example model's decisions in the two groups (100 applicants in each group, the same model, the same threshold 0.5).

Criterion	A	B	Satisfied?
Demographic parity (approval rate)	0.50	0.20	no
Equalized odds (TPR; FPR)	0.80; 0.20	0.50; 0.059	no
Predictive parity (PPV)	0.80	0.80	yes

Table 11.7: The result under three criteria.

The model is fair by the institution's criterion and not fair by the applicant's criteria. Because the base rates differ (0.50 and 0.32), equalized odds and predictive parity cannot hold together except in extreme cases; demographic parity does not hold even under perfect prediction, because the approval rates would then equal the base rates (Chouldechova, 2017; Kleinberg et al., 2017). Which criterion to choose is a value decision. Leaving this decision to developers as if it were a technical detail is also a choice. Who makes the decision, and whether its rationale is written down, are matters for the auditing described in 11.5.4.

11.3.5 A real case: the COMPAS debate

COMPAS is a risk assessment tool used in some courts in the United States. It scores a defendant's risk of reoffending within two years. In 2016 the news organization ProPublica examined the tool using data it obtained from Broward County, Florida (Angwin et al., 2016).

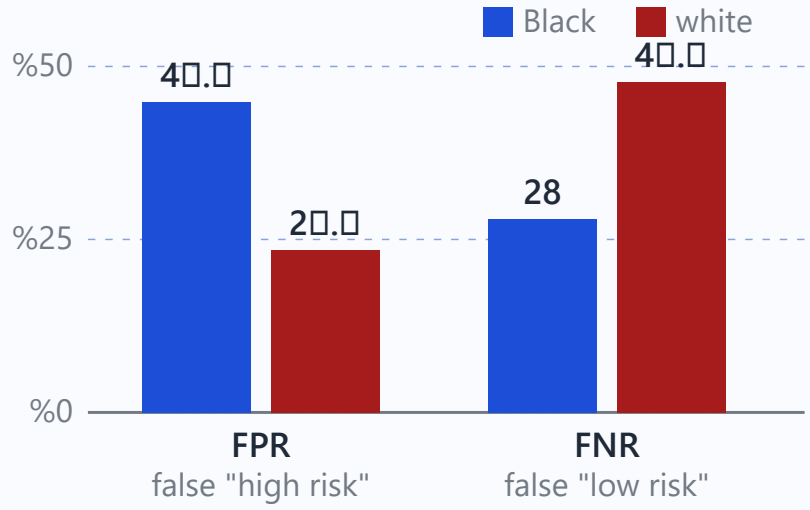


Figure 11.7: Error rates (percent) computed by ProPublica (2016) from the Broward County data. FPR: the share of defendants who did not reoffend but were labeled "high risk" (Black 44.9%, white 23.5%). FNR: the share of those who did reoffend but were labeled "low risk" (Black 28.0%, white 47.7%).

The two sides looked at two different criteria:

- **ProPublica:** 44.9% of Black defendants who did not reoffend had been labeled "high risk"; among white defendants this rate was 23.5%. So the FPR was not equal. For the false negative rate (FNR) the picture reversed: 47.7% of white defendants who did reoffend had been called "low risk", compared with 28.0% of Black defendants. On the basis of these findings, ProPublica argued that the tool was biased against Black defendants.
- **Northpointe (the company that developed the tool):** The same risk score corresponded to a similar reoffending rate in the two groups. That is, the tool was calibrated and satisfied predictive parity. On this ground the company argued that the tool was fair.

Both sides were right in their own computation. Because the base rates in the two groups' reoffending records differ, the two criteria could not hold together (the Chouldechova identity in Theorem 11.1). The debate is really about which definition of fairness to choose. Chouldechova's 2017 paper was written precisely in response to this debate. The lesson is this: instead of a single accuracy score, group-wise error rates should be reported. And when one side says "the model is fair", the first question to ask should be "by which criterion?".

The structure of the case is very similar to the chapter's numerical example. In COMPAS, the group with the higher base rate (the group whose reoffending rate looks higher in the records) also has the higher FPR. This is the direction the identity predicts: when PPV is equal, FPR grows as p grows. In the chapter's example, group A, which has the higher base rate, has an FPR (0.20) larger than B's (0.059).

11.3.6 The fairness–accuracy trade-off

Model accuracy alone does not show on whom the errors are concentrated. In our example the accuracy is 80% in both groups, yet the TPRs are 0.8 and 0.5. Accuracy sums two kinds of errors (FP and FN) and erases the difference between them.

Adding a fairness constraint can increase the total error. This is a **trade-off**: giving up one thing to gain another. But the real choice is subtler than the question "how much accuracy is sacrificed?". The real choice is which errors are acceptable: a wrong rejection or a wrong approval? The same total error can be distributed differently among groups.

A fairness constraint can be added at three points in the development process of a model:

Pre-processing
: The data are changed before the model is trained. For example, the examples of an underrepresented group are given more weight, or more data are collected from that group. The aim is to remove the underrepresentation.

In-processing
: A fairness constraint or penalty term is added to the error measure minimized in training, that is, to the **loss function**. The model tries both to make few errors and to keep the group rates close; to do so it gives up some accuracy.

Post-processing
: Without changing the scores of the trained model, the threshold that makes the decision is adjusted by group (Hardt et al., 2016). This method is simple, but because it explicitly uses group membership in the decision, it can be contested under some legal systems.

Which method and which criterion to choose should be guided by the context of use, legal requirements, regulations, and the views of the affected groups. For example, a credit scoring system that wants to evaluate low-income applicants fairly may count the rejection of members of this group who could repay (low TPR) as the gravest error and give priority to equal opportunity.

11.3.7 Code trace: equal opportunity by randomization

The simplest form of post-processing is this: in group B, instead of approving only those with a score of 0.8, also approve some of those with a score of 0.2 at random. Let the probability of approval be q. The function below computes group B's TPR, FPR, and accuracy for each value of q.

```
def group_b(q):   # approve score 0.2 w.p. q
  TP = 16 + 16 * q
  FP = 4 + 64 * q
  tpr = TP / 32
  fpr = FP / 68
  hits = TP + (68 - FP)    # TP + TN
  return tpr, fpr, round(hits / 100, 3)

for q in [0, 0.2, 0.4, 0.6]:
  tpr, fpr, d = group_b(q)
  print(q, round(tpr, 2), round(fpr, 2), d)
```

The logic of the function comes from Table 11.2. All 20 people with a score of 0.8 are approved; of these 16 repay (TP) and 4 do not (FP). Each of the 80 people with a score of 0.2 is approved with probability q. Of these 80, 16 repay and 64 do not. In expectation, $16 \cdot q$ of those who repay and $64 \cdot q$ of those who do not repay receive an additional approval. Thus $TP = 16 + 16q$ and $FP = 4 + 64q$. The denominator of TPR is all of those in B who repay (32), and that of FPR is all of those who do not repay (68). The correct decisions are the sum of TP and TN; TN is those who do not repay and are not approved: $TN = 68 - FP$. In the code the line `hits = TP + (68 - FP)` does this. So the accuracy is $(16 + 16q + 68 - 4 - 64q)/100 = (80 - 48q)/100$.

q	TP	FP	TPR	FPR	accuracy	selection rate	PPV
0	16	4	0.5	0.06	0.800	0.20	0.80
0.2	19.2	16.8	0.6	0.25	0.704	0.36	0.53
0.4	22.4	29.6	0.7	0.44	0.608	0.52	0.43
0.6	25.6	42.4	0.8	0.62	0.512	0.68	0.38

Table 11.8: Trace of `group_b(q)`. The last two columns are the selection rate and PPV, which are not computed in the code but follow from the same numbers.

The table is read row by row as follows:

1. ***q = 0*:** Only those with 0.8 are approved (today's model). TP = 16, FP = 4. TPR = 16/32 = 0.5; FPR = 4/68 ≈ 0.06; accuracy (16 + 64)/100 = 0.80.
2. ***q = 0.2*:** 20% of the 80 people with 0.2 are approved. TP = 16 + 16 · 0.2 = 19.2; FP = 4 + 64 · 0.2 = 16.8. TPR rises to 0.6 but accuracy falls to 0.704. Why does it fall so much? Only 20% of those with 0.2 repay; of every 5 newly approved people, 4 are among those who will not repay.
3. ***q = 0.4*:** TP = 22.4; FP = 29.6. TPR 0.7; FPR jumps to 0.44; accuracy 0.608.

4. ***q = 0.6*:** TP = 25.6; FP = 42.4. $TPR_B = 25.6/32 = 0.8 = TPR_A$: equal opportunity is achieved. The price: accuracy fell to 0.512 and PPV to 0.38; FPR rose to 0.62.

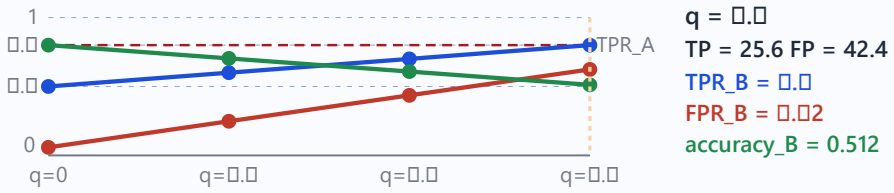


Figure 11.8: The effect of approving those with 0.2 at random with probability q in group B. Blue: TPR_B (rises from 0.5 to 0.8 and reaches A's TPR, the dashed line, at $q = 0.6$). Red: FPR_B (from 0.06 to 0.62). Green: B's accuracy (from 0.8 to 0.512).

Figure 11.8 shows the price plainly. For equal opportunity, 60% of those with 0.2 have been approved at random. Of the 64 who will not repay, about $64 \cdot 0.6 \approx 38$ receive credit. B's FPR (0.62) is more than three times A's (0.2); so equalized odds still does not hold. Predictive parity is broken as well: PPV has fallen from 0.80 to 0.38. It is possible to achieve one criterion by randomization, but this does not increase the model's power to discriminate; it only redistributes the errors.

Example 11.12

If A's TPR were 0.7 instead of 0.8, what would q have to be for equal opportunity in B, and what would B's accuracy be?

1. $TPR_B = (16 + 16q)/32 = 0.5 + 0.5q$.
2. $0.5 + 0.5q = 0.7 \Rightarrow 0.5q = 0.2 \Rightarrow q = 0.4$.
3. Accuracy $(80 - 48 \cdot 0.4)/100 = (80 - 19.2)/100 = 0.608$. It is the same as the $q = 0.4$ row of Table 11.8.

Common mistake

- **"A better model satisfies all the criteria."** When base rates differ, this happens only with perfect prediction. Improving the model can reduce the size of the conflict but does not remove it.
- **Writing p instead of $p/(1 - p)$ in the identity:** The factor is not a probability but an odds ratio ("the ratio of those who repay to those who do not"). At $p = 0.5$ it is 1, not 0.5.
- **Looking for the price of randomization only in accuracy:** As q grows, accuracy falls, but the really important change is the change in FPR and PPV. While one criterion is achieved, the others can move even further away.

- **Confusing balance with calibration:** Calibration asks "what is the actual outcome of people whose score is known?"; balance asks "what is the score of people whose actual outcome is known?". The direction of the condition is reversed.

11.4 Fair division

The second face of fairness is the distribution of a common resource among people. Dividing an inheritance, distributing the processing time of a shared machine among users, parceling out a piece of land, or assigning courses to students are problems of this kind. What is measured here is not a model's error rates but what each person receives in their own eyes. The resource can be divisible (a cake) or consist of indivisible goods; the notions of fairness and the algorithms that achieve them differ in the two cases.

11.4.1 The cake-cutting model

A divisible resource is traditionally described by a cake. The cake is the interval $[0, 1]$ on the number line. This interval can represent time (the hours of a day), land (a stretch of coast), or processing time. The cake is to be divided among n people.

People value different parts of the cake differently. One likes the chocolate part, another the vanilla part; one wants the morning hours, another the evening hours. This difference is described by a value function.

Definition 11.9

In the **cake-cutting** problem, each person i has a **value function** v_i. $v_i(X)$ is the value of piece X in the eyes of person i. The whole cake is worth 1 to every person: $v_i([0, 1]) = 1$. Values are additive: the value of the union of two disjoint pieces is the sum of their values. X_i is the piece that person i receives.

An algorithm does not know people's value functions in full; knowing them would mean an infinite amount of information. Instead it asks people questions. In the standard model proposed by Robertson and Webb there are two kinds of question:

- **Evaluation query:** "What value do you give the piece $[a, b]$?"
- **Cut query:** "Where does the piece that starts at a and has value α in your eyes end?"

This model is called the **Robertson–Webb query model**. The efficiency of an algorithm is measured by how many queries it asks. This measure is like counting how many comparisons an algorithm makes in computer science.

There are two basic ways of saying that an allocation is "fair".

Definition 11.10

An allocation $(X_1, \ldots, X_n)$ is:

- **proportional** if everyone receives at least $1/n$ in their own eyes: $v_i(X_i) \geq 1/n$ for every i;
- **envy-free** if nobody prefers another's piece to their own: $v_i(X_i) \geq v_i(X_j)$ for every i and j.

In both definitions the evaluation is made with each person's *own* value function. That is, fairness is defined not by an outside referee but by the people's own preferences. Ali and Ayşe can receive pieces of different sizes from the same cake, and the allocation can still be fair in both their eyes.

Proportionality is a "lower bound" condition: the person looks only at their own share. Envy-freeness is a "comparison" condition: the person compares their own share with everyone else's. When the cake is fully allocated, envy-freeness implies proportionality. The reason is simple: an envy-free person finds their own piece at least as valuable as each of the n pieces; since the total value of the n pieces is 1 in their eyes, the value of the largest piece is at least $1/n$. The reverse is generally not true (11.4.5).

From the viewpoint of game theory, people may be strategic: they may try to get a bigger piece by giving false answers to the algorithm's questions. A good protocol gives every person who behaves honestly a guarantee, whatever the others do. This guarantee is stronger than a notion like the Nash equilibrium, which assumes that everyone is rational. A person can secure their share without making any assumption about the others' behavior.

11.4.2 Cut and choose step by step

For two people, the oldest known fair division method is the rule children use when sharing a cake: one cuts, the other chooses.

Definition 11.11

The **cut-and-choose** method consists of two steps:

1. The **cutter** divides the cake into two pieces of equal value in their own eyes.
2. The **chooser** takes whichever of the two pieces has the greater value in their own eyes; the remaining piece goes to the cutter.

The example cake consists of two halves: the left half $[0, 0.5]$ is vanilla and the right half $[0.5, 1]$ is chocolate. Ali values every part of the cake equally. Ayşe likes chocolate 3 times as much as vanilla: $v_{Ayşe}(\text{vanilla}) = 1/4$, $v_{Ayşe}(\text{chocolate}) = 3/4$. Within each half the value is spread uniformly; that is, half of a half is worth half the value of that half.

Example 11.13

1. **Ali cuts.** He makes two equal pieces in his own eyes. Since Ali's value is uniform, he cuts at $x = 1/2$. Each piece is worth 1/2 to Ali.
2. **Ayşe evaluates.** The left piece is pure vanilla, so in her eyes it is 1/4; the right piece is pure chocolate, so it is 3/4.
3. **Ayşe chooses.** She takes the piece with the greater value, the right one (chocolate). The left piece goes to Ali.
4. **Result.** Proportionality: Ali 1/2 ≥ 1/2, Ayşe 3/4 ≥ 1/2 ✓. Envy-freeness: Ali values his own piece at 1/2 and Ayşe's at 1/2 (1/2 = 1/2, he does not envy); Ayşe values her own piece at 3/4 and Ali's at 1/4 (3/4 > 1/4) ✓.

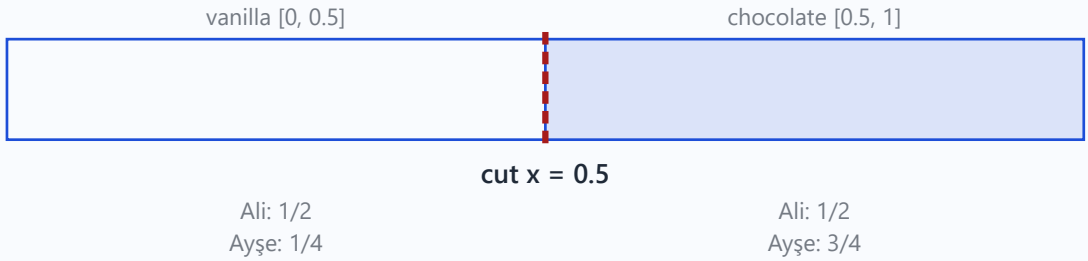


Figure 11.9: Cut and choose, with Ali cutting. The cut is at $x = 0.5$. The left piece is worth 1/2 to Ali and 1/4 to Ayşe; the right piece is worth 1/2 to Ali and 3/4 to Ayşe. Ayşe chooses the right one.

If the roles are swapped, the result changes too. This time let Ayşe cut and Ali choose. Ayşe must make two equal pieces in her own eyes. Since the whole left half (vanilla) is worth only 1/4 to her, the cut must be inside the chocolate. If the cut point is $x > 1/2$, the value of the left piece to Ayşe is $1/4 + (3/4) \cdot (x - 1/2)/(1/2) = 1/4 + 1.5 \cdot (x - 1/2)$. Setting this equal to 1/2 gives $1.5 \cdot (x - 1/2) = 1/4$, that is, $x - 1/2 = 1/6$ and $x = 2/3$.

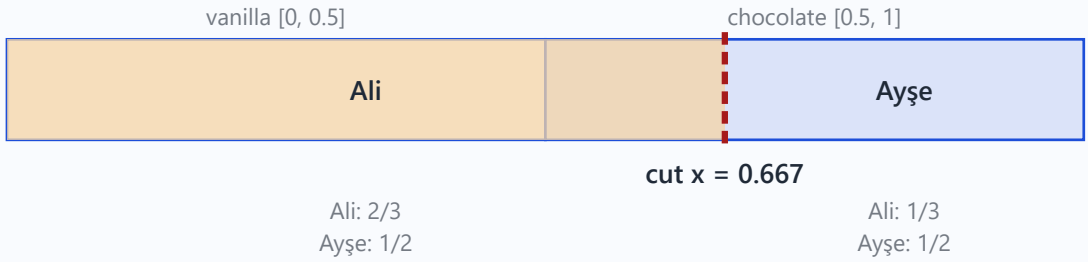


Figure 11.10: When the roles are swapped: Ayşe cuts at $x = 2/3 \approx 0.667$ (1/2–1/2 in her own eyes). Ali chooses the left piece and receives 2/3 in his own eyes; Ayşe receives 1/2.

In Figure 11.10 Ali's choice is easy: the left piece is worth 2/3 to him and the right piece 1/3. Ali gets 2/3, and Ayşe gets exactly 1/2 in her own eyes. Looking at the two cases together, a pattern appears: Ali gets 2/3 when choosing and 1/2 when cutting; Ayşe gets 3/4 when choosing and 1/2 when cutting. The chooser is better off than the cutter. The cutter receives only the guarantee (1/2); the chooser collects the gain from the difference between the two people's valuations.

11.4.3 Code trace: a cut query by bisection

We found Ayşe's cut point (2/3) above with a formula. An algorithm, however, does not know the formula of Ayşe's value function; it can only ask Robertson–Webb queries. The answer to a cut query can be found approximately, using only evaluation queries, by the **bisection** method. The code below implements this idea.

```
def v_ayse(x):          # value of [0, x]
  if x <= 0.5: return x / 2
  return 0.25 + 1.5 * (x - 0.5)
def cut_point(v, target, steps):
  lo, hi = 0.0, 1.0
  for _ in range(steps):
    mid = (lo + hi) / 2
    if v(mid) < target: lo = mid
    else: hi = mid
  return (lo + hi) / 2
print(round(cut_point(v_ayse, 0.5, 8), 3))
```

`v_ayse(x)` gives the value of the piece *[0, x]* to Ayşe. When *x ≤ 0.5* the piece is entirely vanilla and its value is *x/2* (all the vanilla, that is, length 0.5, is worth 1/4). When *x > 0.5*, all the vanilla (0.25) plus part of the chocolate is taken; each unit of length of chocolate is worth *(3/4)/(1/2) = 1.5*.

The function `cut_point(v, target, steps)` knows that the point it is looking for lies in the interval *[lo, hi]*. At the start *lo = 0* and *hi = 1*. At each step it asks about the middle of the interval. If the value at the middle is less than the target, the cut point is further to the right and `lo` moves to the middle; otherwise the cut point is at the middle or to its left and `hi` moves to the middle. Each step halves the interval. For this method to work it suffices that the value function is non-decreasing: the value of *[0, x]* does not decrease as the piece grows.

query	mid	v(mid)	comparison	new [lo, hi]	width
1	0.5	0.25	< 0.5 ⇒ lo = mid	[0.5, 1]	1/2
2	0.75	0.625	≥ 0.5 ⇒ hi = mid	[0.5, 0.75]	1/4
3	0.625	0.4375	< 0.5 ⇒ lo = mid	[0.625, 0.75]	1/8
4	0.6875	0.5312	≥ 0.5 ⇒ hi = mid	[0.625, 0.6875]	1/16
5	0.6562	0.4844	< 0.5 ⇒ lo = mid	[0.6562, 0.6875]	1/32
6	0.6719	0.5078	≥ 0.5 ⇒ hi = mid	[0.6562, 0.6719]	1/64

query	mid	v(mid)	comparison	new [lo, hi]	width
7	0.6641	0.4961	< 0.5 ⇒ lo = mid	[0.6641, 0.6719]	1/128
8	0.668	0.502	≥ 0.5 ⇒ hi = mid	[0.6641, 0.668]	1/256

Table 11.9: Trace of `cut_point(v_ayse, 0.5, 8)`. Each row is one evaluation query.

After eight queries the width of the interval is $1/2^8 = 1/256$. The function returns the middle of the interval, ≈ 0.666. The true cut is $2/3 \approx 0.6667$; the error is about 0.00065. If Ayşe were the cutter, she would cut here.

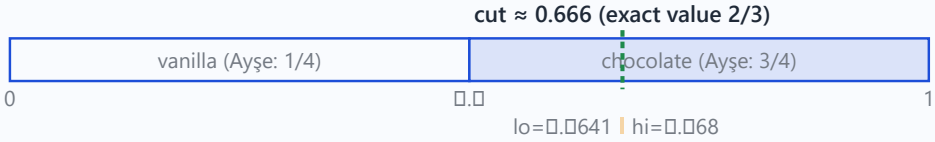


Figure 11.11: After eight queries the search interval narrows to *[0.6641, 0.668]*; the midpoint is ≈ 0.666 and the true cut is 2/3.

How the error shrinks can be written in general. After k queries the width of the interval is 2^{-k}. The true point lies inside this interval and the function returns the middle of the interval; so the error is at most half the interval, that is, $2^{-(k+1)}$. For the error always to stay below 0.001 we need $2^{-(k+1)} < 0.001$. Since $2^{10} = 1024 > 1000$, $k + 1 = 10$, that is, $k = 9$ queries suffice; at $k = 8$ the bound is $2^{-9} \approx 0.00195$ and gives no guarantee. For one more decimal digit of accuracy, about three or four more queries are needed.

Example 11.14

When `cut_point(v_ayse, 0.5, 3)` is called, what is *[lo, hi]* at the end of the loop?

1. Mid 0.5: $v = 0.25 < 0.5 \Rightarrow lo = 0.5$.
2. Mid 0.75: $v = 0.625 \geq 0.5 \Rightarrow hi = 0.75$.
3. Mid 0.625: $v = 0.4375 < 0.5 \Rightarrow lo = 0.625$.

The interval is *[0.625, 0.75]* and the returned value is 0.6875. This is the same as the first three rows of Table 11.9.

11.4.4 Why is cut and choose fair?

The reason cut and choose is fair becomes clear when one looks at the guarantees of the two roles separately.

The cutter's guarantee. The cutter does not know which piece they will end up with; the chooser takes whichever they want. If the cutter thinks of the worst case, they are left with the smaller of the two pieces. So the cutter's guarantee is *min(v(left), v(right))*. As the cut point moves to the right, the value of the left piece rises and that of the right piece falls; the smaller of the two values is largest when the two are equal. This point is exactly the cutter's own halfway point, and the guarantee is 1/2. Cutting honestly, that is, making two equal pieces, is the strategy that maximizes the worst-case gain, the **maximin** strategy.

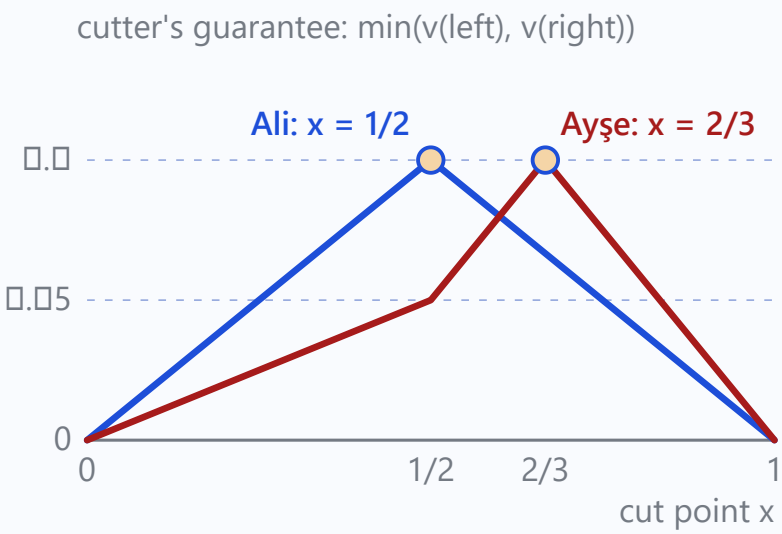


Figure 11.12: The cutter's guarantee *min(v(left), v(right))* as a function of the cut point *x*. For Ali (blue) the peak is at *x = 1/2*, for Ayşe (red) at *x = 2/3*; both peaks have height 0.5.

Figure 11.12 shows this "roof" shape. Ali's curve is smooth, because his value is spread uniformly. The left arm of Ayşe's curve rises slowly up to 1/2 (vanilla is of little value) and then rises fast. The peak of both curves is 0.5: whatever anyone's value function is, an honest cutter guarantees 1/2.

The chooser's guarantee. The chooser takes the larger of the two pieces in their own eyes. Since the values of the two pieces sum to 1, the larger is at least 1/2. Moreover, the piece they choose is not less valuable than the remaining piece. These two properties hold however the cut is made; even if the cutter does not behave honestly, the chooser receives a share that is both proportional and envy-free.

The value of information. If the cutter knows the chooser's preference, they can shift the cut accordingly and receive more than 1/2; but their guarantee is still 1/2. The example below shows this with numbers.

Example 11.15

Ali cuts, and he knows that Ayşe likes chocolate 3 times as much. If Ayşe finds the two pieces equally valuable, she chooses the chocolate (right) piece. Where does Ali cut, and what does he gain in his own eyes?

1. Ayşe chooses the right piece as long as the value of the left piece in her eyes is at most 1/2.
2. If the cut is at $x \geq 1/2$, the value of the left piece to Ayşe is $1/4 + 1.5\,(x - 1/2)$. The condition $1/4 + 1.5\,(x - 1/2) \leq 1/2$ means $x \leq 2/3$.
3. Since Ali wants to take the left piece, he makes it as large as possible: he cuts at $x = 2/3$.
4. Ali receives $2/3 > 1/2$ in his own eyes; Ayşe again receives 1/2.

Information gives the cutter an advantage. But if Ali's information is wrong (for example, if Ayşe actually likes vanilla), this shifted cut can drop Ali below 1/2. The value of the honest cut is that it gives a guarantee that requires no information.

That cut and choose is both proportional and envy-free for two people is not a coincidence; for two people these two notions are the same.

Theorem 11.4 (Proportionality and envy-freeness for two people)

If the cake is fully divided between two people, then for one person proportionality and envy-freeness are equivalent: $v(X_1) \geq 1/2 \Leftrightarrow v(X_1) \geq v(X_2)$.

Proof. Since the cake is fully divided, $v(X_1) + v(X_2) = 1$. ($\Rightarrow$) Let $v(X_1) \geq 1/2$. Then $v(X_2) = 1 - v(X_1) \leq 1/2 \leq v(X_1)$; the person does not envy the other. ($\Leftarrow$) Let $v(X_1) \geq v(X_2) = 1 - v(X_1)$. Adding $v(X_1)$ to both sides gives $2\,v(X_1) \geq 1$, that is, $v(X_1) \geq 1/2$. □

For three or more people this equivalence breaks down: envy-freeness still implies proportionality, but a proportional allocation can contain envy. Envy-freeness becomes the stronger condition.

11.4.5 Proportional but not envy-free: a three-person example

Three people (Ali, Ayşe, Cem) will share three pieces of a cake (X, Y, Z) that have already been cut. The table below shows the value each person gives to each piece; every row sums to 1.

person	piece X	piece Y	piece Z	receives	≥ 1/3?
Ali	**0.40**	0.30	0.30	X: 0.40	yes
Ayşe	0.20	**0.35**	0.45	Y: 0.35	yes
Cem	0.30	0.30	**0.40**	Z: 0.40	yes

Table 11.10: The values three people give to three pieces, and the allocation Ali → X, Ayşe → Y, Cem → Z.

Example 11.16

1. **Proportionality.** $1/3 \approx 0.333$. Ali receives 0.40, Ayşe 0.35, Cem 0.40; all three are above 0.333. The allocation is proportional.

2. **Ali's envy.** For Ali, Y and Z are 0.30 and his own X is 0.40. Ali envies nobody.
3. **Cem's envy.** For Cem, X and Y are 0.30 and his own Z is 0.40. Cem envies nobody.
4. **Ayşe's envy.** For Ayşe, Cem's piece Z is worth 0.45 and her own Y is worth 0.35. *0.45 > 0.35*: Ayşe envies Cem.

The allocation is proportional but not envy-free. Since Ayşe receives more than 1/3, she cannot say "I was treated unfairly", but she would like to swap with Cem.

There is a simple fix in this example: if Ayşe and Cem swapped pieces, Ayşe would get 0.45 and Cem would get 0.30. But this time Cem would fall below 1/3 and the allocation would no longer be proportional. Finding an envy-free allocation is generally much harder than finding a proportional one. This difference is reflected in the complexity of the known algorithms:

Last diminisher
: Finds a proportional allocation for n people (11.4.6).

Selfridge–Conway
: Finds an envy-free allocation for 3 people, with at most 5 cuts.

Aziz–Mackenzie (2016)
: The first algorithm that finds an envy-free allocation for n people and terminates after a finite number of queries. But the upper bound on the number of queries is enormous; it is a function of n that grows like a tower of exponents. The algorithm is important in theory and is not used in practice.

11.4.6 Last diminisher

One way to extend cut and choose to more than two people is to put the first piece through everyone's inspection. This idea leads to the **last diminisher** method.

Definition 11.12

The **last diminisher** method works for n people as follows:

1. The first person cuts, from the left of the cake, a piece worth $1/n$ in their own eyes.
2. The piece is shown to the other people in turn. A person who finds the piece worth more than $1/n$ in their own eyes trims it (diminishes it) so that it is worth exactly $1/n$ in their own eyes. A person who finds the piece worth $1/n$ or less does not touch it ("passes").
3. The person who trimmed the piece last takes it and leaves the game. If nobody trimmed, the piece goes to the first cutter.
4. The remaining cake is divided among the remaining $n - 1$ people with the same method. When two people remain, cut and choose is applied.

To see why the method is proportional it suffices to look at two kinds of people. The person who takes the piece is the one who trimmed it last; in their own eyes they receive exactly *1/n*. For those who do not take the piece, the piece is worth at most *1/n* in their eyes: either it was so from the start or they trimmed it to make it so (if someone after them trimmed it, the piece became even smaller). So the remaining cake is worth at least $1 - 1/n = (n - 1)/n$ to them. If the remaining $n - 1$ people divide this cake proportionally, each receives at least $((n - 1)/n) \cdot (1/(n - 1)) = 1/n$.

In the example there are three people and the cake is divided into four equal quarters. Within each quarter the value is spread uniformly. Ali's value is uniform; Ayşe likes the right side, and Cem likes the left.

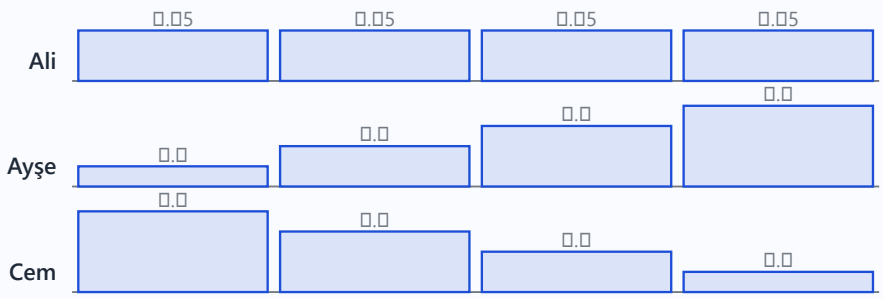


Figure 11.13: The last diminisher example. The bars are each person's value per quarter (every row sums to 1): Ali 0.25–0.25–0.25–0.25; Ayşe 0.1–0.2–0.3–0.4; Cem 0.4–0.3–0.2–0.1.

A tool for the computations is this: since the length of a quarter is 1/4, in a quarter of value *w* a unit of length is worth *4w*. For example, in Cem's first quarter a unit of length is worth $4 \cdot 0.4 = 1.6$; the piece *[0, 0.1]* is worth $0.1 \cdot 1.6 = 0.16$ to him.

Example 11.17

1. **Ali cuts.** A piece worth 1/3 in his own eyes: *[0, 1/3]*. Since Ali's value is uniform, length and value are the same.
2. **Ayşe looks.** The piece contains the whole first quarter (0.1) and a length of $1/3 - 1/4 = 1/12$ from the second quarter. In the second quarter a unit of length is worth $4 \cdot 0.2 = 0.8$. The value is $0.1 + (1/12) \cdot 0.8 = 0.1 + 0.0667 = 1/6$. $1/6 < 1/3$; Ayşe does not touch it.
3. **Cem looks.** The value is $0.4 + (1/12) \cdot 1.2 = 0.4 + 0.1 = 1/2$. $1/2 > 1/3$; Cem trims the excess. Since the first quarter alone is $0.4 > 1/3$, the cut is inside the first quarter: $1.6 \cdot x = 1/3 \Rightarrow x = 1/(3 \cdot 1.6) = 5/24 \approx 0.208$. The piece is now *[0, 5/24]* and is worth exactly 1/3 in Cem's eyes.

4. **Cem takes it.** Cem is the last to trim the piece; he takes it and leaves. For Ali, who did not trim, the piece is *5/24 ≤ 1/3*, and for Ayşe it is *(5/24) · 0.4 = 1/12 ≤ 1/3*. The remaining cake is worth at least 2/3 in the eyes of both.
5. **Cut and choose between the two on the remainder *[5/24, 1]*.** The remaining cake is worth *1 − 5/24 = 19/24* to Ali. Ali divides it into two equal pieces in his own eyes: half is *19/48*, and the cut is at *5/24 + 19/48 = 10/48 + 19/48 = 29/48 ≈ 0.604*.
6. **Ayşe chooses.** The right piece *[29/48, 1]* for Ayşe: a length of *3/4 − 29/48 = 7/48* from the third quarter (unit value 1.2) and the whole fourth quarter (0.4). The value is *(7/48) · 1.2 + 0.4 = 0.175 + 0.4 = 0.575 = 23/40*. The left piece *[5/24, 29/48]* is worth *41/120 ≈ 0.342* to her. Ayşe chooses the right one.

Result: Cem receives *[0, 5/24]*, worth 1/3 in his own eyes; Ali receives *[5/24, 29/48]*, worth *19/48 ≈ 0.396*; Ayşe receives *[29/48, 1]*, worth *23/40 = 0.575*. Everyone received at least 1/3 in their own eyes; the allocation is proportional.

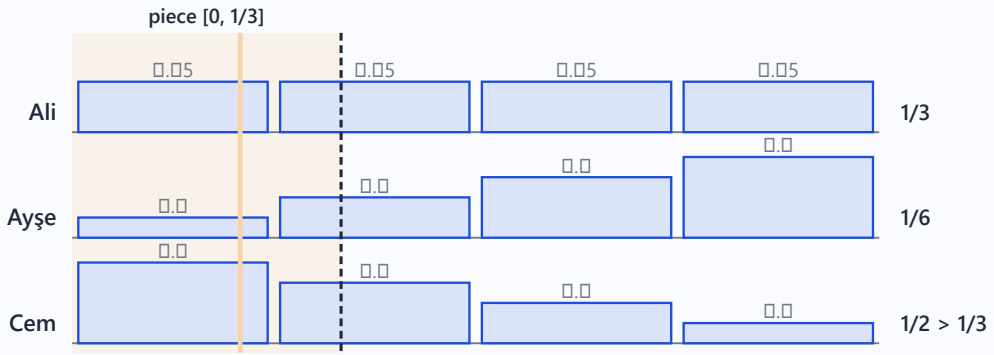


Figure 11.14: The piece *[0, 1/3]* cut by Ali (shaded area). The piece is worth 1/3 to Ali, 1/6 to Ayşe, and 1/2 to Cem. Cem pulls the cut in to 5/24 (orange line).

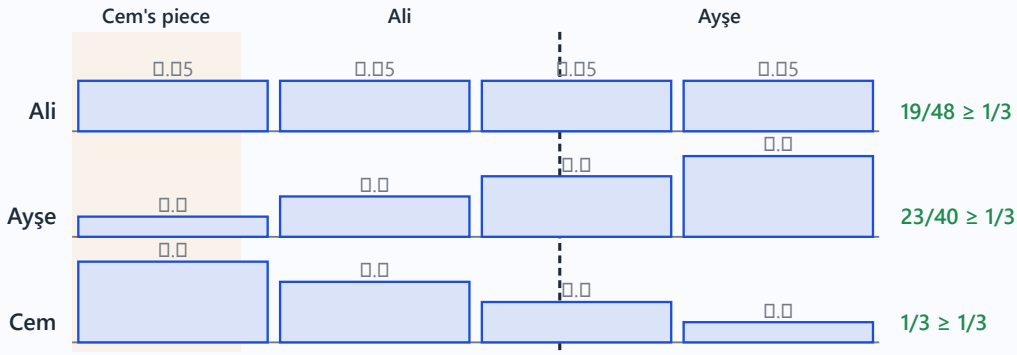


Figure 11.15: The outcome of the last diminisher. Cem receives *[0, 5/24]*, Ali *[5/24, 29/48]*, Ayşe *[29/48, 1]*. The numbers on the right are the value each person gives to their own piece: 19/48, 23/40, and 1/3; all ≥ 1/3.

Figure 11.15 shows that the allocation is proportional. But the allocation is not envy-free. Cem gives Ali's piece *[5/24, 29/48]* the following value: from the first quarter a length of *1/24* (unit 1.6) ⇒ 1/15; the whole second quarter ⇒ 0.3; from the third quarter a length of *5/48* (unit 0.8) ⇒ 1/12. The total is *1/15 + 3/10 + 1/12 = 9/20 = 0.45*. Cem's own piece is 1/3 ≈ 0.333: Cem envies Ali. The last diminisher guarantees only proportionality.

The number of queries of the method can also be computed. In the first round a piece is shown to each of the n people; in the second round to $n - 1$ people, and so on. The total is about $n + (n - 1) + \dots + 2 \approx n^2/2$ queries; that is, $O(n^2)$.

Example 11.18

If Cem's values were like Ayşe's (0.1, 0.2, 0.3, 0.4), who would take the first piece?

Ali again cuts *[0, 1/3]*. For Ayşe this piece is 1/6 (the computation above). Since Cem's values are the same, it is 1/6 for him as well. Both find *1/6 < 1/3* and do not touch it. Since nobody trims, the piece goes to the first cutter, that is, Ali.

11.4.7 Moving knife and Even–Paz

The last diminisher method also has a continuous version. Dubins and Spanier (1961) describe it with the image of a knife.

Definition 11.13

In the **moving knife** (Dubins–Spanier) method, a knife slides slowly from the left of the cake to the right. When the piece to the left of the knife reaches the value *1/n* in someone's eyes, that person says "stop!", takes the piece, and leaves. The knife continues with the rest; the last person takes what remains.

Let us run the method with the values of the last diminisher example (per quarter: Ali 0.25; Ayşe 0.1–0.4; Cem 0.4–0.1). For each person the answer to "when does the left piece become worth 1/3?" is computed; the person with the smallest answer says "stop" first.

person	Round 1: $v([0, x]) = 1/3$	Round 2: $v([5/24, x]) = 1/3$	receives
Cem	$x = 5/24 \approx 0.21$ (stop!)	—	*[0, 5/24]*: 1/3
Ali	$x = 1/3 \approx 0.33$	$x = 13/24 \approx 0.54$ (stop!)	*[5/24, 13/24]*: 1/3
Ayşe	$x = 19/36 \approx 0.53$	$x = 43/72 \approx 0.60$	*[13/24, 1]*: 13/20

Table 11.11: The moving knife. Each cell is the knife position at which that person would say "stop".

Example 11.19

1. **Round 1, Cem:** In the first quarter the unit value is 1.6; $1.6\,x = 1/3 \Rightarrow x = 5/24$.
2. **Round 1, Ali:** Uniform value; $x = 1/3$.
3. **Round 1, Ayşe:** The first two quarters give her 0.1 + 0.2 = 0.3, which is less than 1/3. In the third quarter the unit value is 1.2: $0.3 + 1.2\,(x - 1/2) = 1/3 \Rightarrow x - 1/2 = 1/36 \Rightarrow x = 19/36$.
4. The smallest position is Cem's (5/24). When the knife gets there, Cem stops it and takes $[0, 5/24]$. The first piece is the same as in the last diminisher: Cem stops the cut first.
5. **Round 2, Ali:** $5/24 + 1/3 = 5/24 + 8/24 = 13/24$.
6. **Round 2, Ayşe:** $[5/24, 1/4]$ gives her $(1/24) \cdot 0.4 = 1/60$ and the second quarter gives 0.2; up to 1/2 the total is $13/60 \approx 0.217$. The shortfall $1/3 - 13/60 = 7/60$ is made up in the third quarter with a length of $(7/60)/1.2 = 7/72$: $x = 1/2 + 7/72 = 43/72$.
7. Ali's position (13/24 ≈ 0.542) is smaller than Ayşe's (43/72 ≈ 0.597). Ali stops the knife and takes $[5/24, 13/24]$. Ayşe takes the rest: $[13/24, 1]$, whose value is $(3/4 - 13/24) \cdot 1.2 + 0.4 = (5/24) \cdot 1.2 + 0.4 = 0.25 + 0.4 = 13/20$.

Everyone receives at least 1/3; the allocation is proportional. It is not envy-free: Cem values Ali's piece $[5/24, 13/24]$ at $1/15 + 0.3 + 1/30 = 2/5 > 1/3$.

The moving knife is the "continuous-time" form of the last diminisher; to run it as an algorithm, each person's "stop" point is found with a cut query. In each round a cut query is asked of everyone remaining; the total number of queries is again $O(n^2)$. Even and Paz (1984) find a proportional allocation with far fewer queries, $O(n \log n)$, by dividing the cake in two at every step. The idea is this: everyone marks the halfway point of the cake in their own eyes; the cake is cut at the median of the marks; half of the people share the left piece and half the right piece among themselves with the same method. Since the group is halved at every step, there are about $\log_2 n$ levels, and n queries are asked at each level.

11.4.8 Indivisible goods: round-robin and EF1

A cake can be cut wherever one wants; but a car, a house, or a course seat is indivisible. With **indivisible goods** each item is given entirely to one person. The set of items a person receives is called a **bundle**. Values are assumed to be additive: the value of a bundle is the sum of the values of the items in it.

With indivisible goods, envy-freeness is not always possible. The simplest example is sharing a single car between two siblings. The sibling who gets the car does not envy the other, but the other, who gets nothing, envies the one with the car. Whichever allocation is chosen, someone envies. So a weakened notion of fairness is needed.

Definition 11.14

An allocation is **envy-free up to one good** (EF1) if, in every case of envy, removing a suitable item from the envied bundle eliminates the envy. That is, if i envies j, then for some $g \in X_j$ we have $v_i(X_i) \geq v_i(X_j \setminus \{g\})$.

Round-robin: the people go around in a fixed order, and in each turn each takes the most valuable remaining item in their own eyes.

EF1 means "even if there is envy, it stems from the luck of a single item". In the single-car example, if the car is removed from the bundle of the sibling who has it, an empty bundle remains; the other sibling's envy disappears. This allocation is EF1.

In the example there are two people and four items. Ali starts.

person	g1	g2	g3	g4
Ali	7	5	3	1
Ayşe	8	6	1	1

Table 11.12: Four indivisible items and the values of two people.

Example 11.20

1. **Ali** takes g1 (7). Ali: own 7. Ayşe: own 0, Ali's bundle 8 in her eyes.
2. **Ayşe** takes g2 (6) from the remaining. Ayşe: own 6, Ali's bundle 8. Ali: own 7, Ayşe's bundle 5.
3. **Ali** takes g3 (3). Ali: own 10, Ayşe's bundle 5. Ayşe: own 6, Ali's bundle 8 + 1 = 9.
4. **Ayşe** takes g4 (1). Ayşe: own 7, Ali's bundle 9. Ali: own 10, Ayşe's bundle 5 + 1 = 6.

Result: Ali {g1, g3} = 10, Ayşe {g2, g4} = 7. Ali does not envy (10 ≥ 6). Ayşe, however, values Ali's bundle at 9 and receives 7: she envies! But if g1 is removed from Ali's bundle, 1 remains in Ayşe's eyes and $1 \leq 7$: the allocation is EF1.

When values are non-negative and additive and each person, on their turn, chooses the item most valuable in their own eyes among those remaining, it can be shown that round-robin always produces an EF1 allocation. For two people the reason is short. Each of Ayşe's picks comes before Ali's next pick and takes the most valuable item remaining at that moment. So Ayşe's k-th pick is, in her eyes, worth no less than Ali's $(k + 1)$-th pick. When Ali's first pick (g1) is removed, the rest of Ali's items are matched one to one with Ayşe's items, and in each pair Ayşe values her own item at least as much as the other. In the example, Ayşe's g2 (6) is worth more than Ali's g3 (1 to Ayşe). The person who starts (Ali) never envies, because in every round he picks before the other.

Example 11.21

If Ayşe had started in the round-robin (with the same values), who would take what? If a person finds two items equally valuable, they take the one with the smaller number.

1. Ayşe takes g1 (8).
2. Ali takes g2 (5) from the remaining.
3. For Ayşe, g3 and g4 are equal (1 and 1); she takes g3, the one with the smaller number.
4. Ali takes g4 (1).

Ali {g2, g4} = 6, Ayşe {g1, g3} = 9. Ali values Ayşe's bundle at 7 + 3 = 10: he envies. But when g1 is removed, 3 remains and $3 \leq 6$: it is EF1 again. This time Ayşe, who starts, does not envy.

11.4.9 Back to the game: greedy picking is not a best response

Let us return to the opening Fair Division Game (11.1.1). The values of the items (P1 / P2) are: Cake 5/1, Book 4/9, Headphones 3/8, Ball 1/1. P1 starts and P2 takes, in each round, the remaining item that is most valuable in its own eyes. Let us compare two plans for P1.

P1's plan	Picking order	P1	P2
always the most valuable	Cake, Book, Headphones, Ball	8	10
Book first	Book, Headphones, Cake, Ball	9	9

Table 11.13: P1's two plans and their outcomes.

Example 11.22

1. **Plan 1 (greedy).** P1 takes its most valuable item, the Cake (5). P2 takes the Book (9). From the remaining, P1 takes the Headphones (3) and P2 takes the Ball (1). P1: 5 + 3 = 8. P2: 9 + 1 = 10.
2. **Plan 2 (Book first).** P1 takes the Book (4). From the remaining, P2 takes its most valuable item, the Headphones (8). P1 takes the Cake (5) and P2 takes the Ball (1). P1: 4 + 5 = 9. P2: 8 + 1 = 9.
3. **Why does it work?** P2 gives the Cake a value of only 1; the Cake will not run away if it waits. What P2 really wants is the Book. If P1 takes the Book first, it takes P2's favorite item away, and it can still take the Cake in the next round.

When all the picking orders of the four items are tried, the largest value P1 can get against this P2 is 9; Plan 2 is the best response. In round-robin, taking the most liked item in every round is not always the best strategy.

Let us also look at what happens in terms of fairness. In the second plan, P2 values P1's bundle {Book, Cake} at $9 + 1 = 10$; its own bundle {Headphones, Ball} is worth 9. P2 envies. If the Book is removed, $1 \leq 9$: the allocation is EF1 again. The guarantee from theory is this: if both sides always take their favorite, round-robin produces EF1. Once one side behaves strategically, this guarantee no longer comes from the theory; in this example EF1 still holds, but there is no guarantee that it will hold in every case.

This observation has also been seen in real life. Harvard Business School used to allocate courses to students with a method based on a round-robin draft. Budish and Cantillon (2012) studied the data of this system. Students made exactly this kind of strategic choice: they held back courses that were unpopular but valuable to themselves and moved up the courses that others also wanted. This behavior caused a loss of welfare. Even a fair protocol is open to strategic players: the EF1 guarantee holds if everyone takes their favorite when their turn comes.

11.4.10 Three social welfare criteria

To choose among several allocations, one has to rank them with a number. This number is called a social welfare criterion. Three criteria are common.

Definition 11.15

- **Utilitarian welfare**: the social welfare $SW = \Sigma\ v_i(X_i)$, that is, the sum of the values, is maximized. This criterion measures efficiency.
- **Egalitarian welfare**: the smallest $v_i(X_i)$ is maximized (maximin). The share of the worst-off person is protected.
- **Nash social welfare**: the product of the values $\Pi\ v_i(X_i)$ is maximized. For two people it is $v_{Ali} \times v_{Ayşe}$.

The Nash social welfare stands between the two extremes. In a sum, one person's large gain easily offsets another's small loss; in a product, however, if any one value approaches zero, it drags the whole product to zero. For this reason the product rewards balanced allocations. The Nash social welfare has one more property: it is unaffected by a change of scale. If all of Ali's values are multiplied by 10, the product of every allocation is multiplied by 10; the ranking does not change and the best allocation stays the same. With a sum, in contrast, if Ali's values are scaled up enough, all the items Ali values go to him. Since comparing people's value scales is often meaningless, this property matters.

With the values in Table 11.12, three of the 16 possible allocations can be compared.

Allocation	Ali	Ayşe	sum	minimum	product	envy-free?
Round-robin: Ali {g1, g3}, Ayşe {g2, g4}	10	7	17	7	70	no, EF1
Utilitarian: Ali {g3, g4}, Ayşe {g1, g2}	4	14	18	4	56	no, not even EF1

Allocation	Ali	Ayşe	sum	minimum	product	envy-free?
Nash / egalitarian (not unique): Ali {g2, g3, g4}, Ayşe {g1}	9	8	17	8	72	yes

Table 11.14: Comparison of three allocations under the welfare criteria.

Example 11.23

1. **The utilitarian allocation.** To maximize the sum, each item goes to whoever values it more: g1 (8 > 7) and g2 (6 > 5) to Ayşe, g3 (3 > 1) to Ali. For g4 there is a tie (1 = 1); here it is given to Ali. Total 4 + 14 = 18.
2. **The unfairness of the utilitarian allocation.** Ali values Ayşe's bundle {g1, g2} at $7 + 5 = 12$ and receives 4 himself. Even if g1 is removed, $5 > 4$; if g2 is removed, $7 > 4$. Whichever item is removed, the envy persists: it is not even EF1.
3. **The Nash allocation.** Ali {g2, g3, g4} = 5 + 3 + 1 = 9, Ayşe {g1} = 8; product 72. Ali values Ayşe's bundle at 7 (< 9); Ayşe values Ali's bundle at 6 + 1 + 1 = 8 (= 8). Nobody envies.
4. **Not unique.** The allocation Ali {g2, g3}, Ayşe {g1, g4} also gives (8, 9) and its product is again 72. Both allocations make the smallest value 8; both are optimal for Nash and egalitarian welfare.

That in the example the allocation that maximizes the Nash social welfare is also fair is not a coincidence. Caragiannis et al. (2016) showed that such an allocation is always at least EF1.

Theorem 11.5 (Caragiannis et al., 2016)

When values are non-negative and additive, the allocation that maximizes the Nash social welfare is both Pareto optimal and EF1. If the product is 0 for every allocation, first the number of people who receive positive utility is maximized, and then the product of those people's utilities.

This result is also reflected in the title of the paper: "The unreasonable fairness of maximum Nash welfare". The theorem shows that efficiency and fairness need not exclude each other. Maximizing a single number (the product) yields an allocation that can neither be improved without hurting someone nor contains envy beyond one item.

11.4.11 The price of fairness and Pareto efficiency

Two notions are needed to measure how much a fairness constraint takes away from efficiency.

Definition 11.16

- An allocation is **Pareto efficient** (Pareto optimal) if there is no other allocation that improves at least one person without making anyone worse off.
- The **price of fairness** is the ratio of the largest social welfare among all allocations to the largest social welfare among fair allocations. If the ratio is close to 1, fairness is cheap.

With the values of Table 11.12, all 16 allocations can be scanned (each of the four items goes to one of two people: $2^4 = 16$). The results of the scan are as follows. 11 of the 16 allocations are Pareto optimal; round-robin's (10, 7) allocation is one of them. For example, there is no other allocation that gives Ali no less than 10 and Ayşe no less than 7 and raises one of them. The allocations that maximize the sum and those that are envy-free are in the table below.

Allocation	Ali	Ayşe	sum	envy-free?
Ali {g3, g4}, Ayşe {g1, g2}	4	14	18	no
Ali {g3}, Ayşe {g1, g2, g4}	3	15	18	no
Ali {g2, g3, g4}, Ayşe {g1}	9	8	17	yes
Ali {g2, g3}, Ayşe {g1, g4}	8	9	17	yes

Table 11.15: The allocations with the largest sum and the envy-free allocations.

Example 11.24

1. **The largest social welfare:** 18. Two allocations give it; they are identical except for who gets g4 (on g4 the two people's values are equal).
2. **Envy-free allocations:** of the 16 allocations only two are envy-free. Because of the tie on g4, both are optimal for Nash and egalitarian welfare; their sums are 17.
3. **The price of fairness:** $18/17 \approx 1.059$.
4. **Interpretation:** For envy-freeness, $1 - 17/18 = 1/18 \approx 5.6\%$ of the social welfare is sacrificed.

The price of fairness is a ratio like the Price of Anarchy. In the Price of Anarchy, the loss comes from the players behaving selfishly: it is the ratio of the best outcome to the worst equilibrium reached by selfish players. In the price of fairness, the loss comes not from selfishness but from the fairness constraint. In both ratios the denominator holds the welfare "attainable under the constraint" and the numerator the unconstrained best welfare.

In this example the price is small. But the price is not always small. In the divisible cake, with n people the price of proportionality grows roughly in proportion to $\sqrt{n}$ (Caragiannis et al., 2012). As the number of people grows, the obligation to give everyone at least $1/n$ becomes increasingly expensive

compared with leaving the cake to whoever values it most.

Common mistake

- **Evaluating values on a common scale:** Proportionality and envy-freeness are checked with each person's own value function. Whether Ayşe envies Ali is computed with Ayşe's values, not Ali's.
- **"If it is proportional it is envy-free":** This is true only for two people. In the three-person example (Table 11.10) the allocation is proportional but Ayşe envies Cem.
- **Removing any item in EF1:** EF1 asks for *a suitable* item to be removed from the envied person's bundle; usually the item that is most valuable in the envier's eyes is removed. If one removes a random item and the envy persists, it does not follow that the allocation is not EF1.
- **Giving the piece to the first cutter in the last diminisher:** The person who trimmed the piece last takes it. Only if nobody trimmed does the piece go to the first cutter.
- **Finding the cut by length:** The cutter makes equal pieces "in their own eyes". Ayşe's halfway point is not at length 1/2 but at the point 2/3.

11.5 Ethics and governance

Criteria and algorithms describe the measurable face of fairness. But mathematics does not decide by which criterion a model will be evaluated, who may object, or who is responsible if there is an error. These questions extend to ethical principles and the legal framework, to people's strategic response to the algorithm, to auditing, and to how decisions change their own data.

11.5.1 Ethical principles and standards

Although the principles proposed for algorithmic decision systems appear under different names in different institutions, four principles are found in almost all of them.

Transparency and explainability

: **Transparency** means that the decision process is understandable, that is, which data are used for which purpose. **Explainability** means that the reason for a single decision can be explained to the person concerned. A person whose credit is rejected should be able to learn the reason for the decision (for example "debt-to-income ratio too high").

Accountability

: **Accountability** means that it is clear who is responsible for a wrong decision. "The algorithm decided so" does not count as a justification. A person affected by the decision should be able to object and to ask a human to review the decision.

Fairness and equality

The social effects of decisions and their outcomes by group must be taken into account. The criteria computed in this chapter are the measurable counterpart of this principle.

Privacy

Privacy means that personal data are collected to the minimum extent necessary and only for the stated purpose. The need for group information in order to measure fairness can create tension with privacy: an institution that does not collect the protected attribute also cannot measure whether its own model is fair with respect to that attribute.

These principles are also reflected in legal regulations.

Regulation	Content
GDPR (EU, 2018), Article 22	The right to object and to human intervention for decisions based solely on automated processing that significantly affect the person.
EU AI Act (2024)	A risk-based approach: applications such as credit scoring and hiring are in the "high-risk" class and are subject to additional obligations (risk management, data quality, human oversight).
KVKK (Turkey), OECD AI Principles	Rules for processing personal data (Law No. 6698); shared principles for trustworthy artificial intelligence.

Table 11.16: The main regulations and principles on algorithmic decisions.

What the regulations have in common is that the human is not removed entirely from high-impact decisions. The risk-based approach of the EU AI Act also carries a principle of proportionality: a movie recommendation system and a credit scoring system are not given the same obligations; the obligations grow with the magnitude of the possible harm.

11.5.2 Strategic design and incentives

In the earlier sections people were treated as passive data points facing the model. But when an algorithm sets a rule, people respond to it. The people being classified are players too. This viewpoint is called **strategic classification**. The language of game theory can be used directly here: the institution chooses a rule (the leader) and the people give the best response to this rule (the followers).

A general consequence of this response is summarized by **Goodhart's law**: when a measure becomes a target, it ceases to be a good measure. Once click counts start to be used as a measure of success, content creators write clickbait headlines; clicks go up, but what the measure was meant to capture, the interest of the content, does not. The link between the measure and the quality one wants to measure is broken by targeting the measure.

Giving users the right incentives and preventing malicious manipulation (fake reviews, bot accounts) is part of the design. Three cases stand out:

A gameable feature

In a credit score, a feature that is easy to change, such as "the number of credit cards a person holds", changes the score without changing the real risk. The person's ability to pay stays the

same while the score rises. This kind of behavior is called **gaming**.

Recommender systems
: An objective that rewards engagement (watch time, clicks) can conflict with the user's long-term benefit. A better objective is one that rewards the user's long-term benefit; but measuring this benefit is much harder.

Unequal costs
: If the cost of gaming differs among groups, strategic behavior creates a new unfairness. The group with plentiful resources adapts to the rule more easily. The example in 11.5.3 shows this step by step.

The basic question of mechanism design applies here too: honest behavior that follows the rule should be in the individual's interest. In a strategy-proof mechanism, behaving honestly is the best strategy; in a well-designed classifier too, the cheapest way to raise one's score should be to actually become more qualified.

11.5.3 Strategic classification step by step

Hu, Immorlica, and Vaughan (2019) studied the effect of strategic behavior across groups with a simple model. The example below is a numerical version of this model.

The institution accepts those whose score is equal to or greater than the threshold. At first the threshold is 0.6. The true qualification is also measured by this score: people whose true score is 0.6 or above are truly qualified. People can raise their scores at a cost (for example by increasing the number of cards). Group A's budget is 0.15: everyone in A can raise their score by at most 0.15. Group B's budget is only 0.03. Each group has 5 unqualified and 5 qualified people.

group	budget	unqualified (score < 0.6)	qualified (score ≥ 0.6)
A	0.15	0.30; 0.42; 0.50; 0.52; 0.58	0.62; 0.66; 0.71; 0.80; 0.90
B	0.03	0.25; 0.35; 0.48; 0.56; 0.58	0.63; 0.68; 0.72; 0.78; 0.88

Table 11.17: True scores in the strategic classification example.

A person's decision is simple. If the true score is below the threshold and the distance to the threshold is less than or equal to the budget, the person moves the score to exactly the threshold and is accepted. If the distance is greater than the budget, raising the score is of no use; the person does not bear the cost.

Example 11.25

1. **Threshold 0.6, gaming.** In A, the unqualified with a true score of at least $0.6 - 0.15 = 0.45$ can reach the threshold: 0.50, 0.52, and 0.58. 3 unqualified people from A get through. In B at least $0.6 - 0.03 = 0.57$ is needed: only 0.58. 1 unqualified person from B gets through.

2. **The institution responds.** The institution notices the gaming and raises the threshold to *0.6 + 0.15 = 0.75*, taking A's budget into account. The highest score the unqualified can reach is *0.58 + 0.15 = 0.73* in A and *0.58 + 0.03 = 0.61* in B. Both are below 0.75; now no unqualified person can get through.
3. **The qualified members of A.** To get past the threshold, the true score must be at least *0.75 − 0.15 = 0.60*. All five qualified members of A satisfy this (the lowest is 0.62). In A, 5 of 5 get through.
4. **The qualified members of B.** The true score must be at least *0.75 − 0.03 = 0.72*. 0.72, 0.78, and 0.88 get through; 0.63 and 0.68 do not. In B only 3 of 5 get through.

The measure taken against strategic behavior created a new inequality. The measure succeeded in keeping the unqualified out, but its price was paid by the qualified members of group B, whose budget is small. The institution's rule looks the same to the two groups (the same threshold), but the two groups' power to adapt to this rule differs.

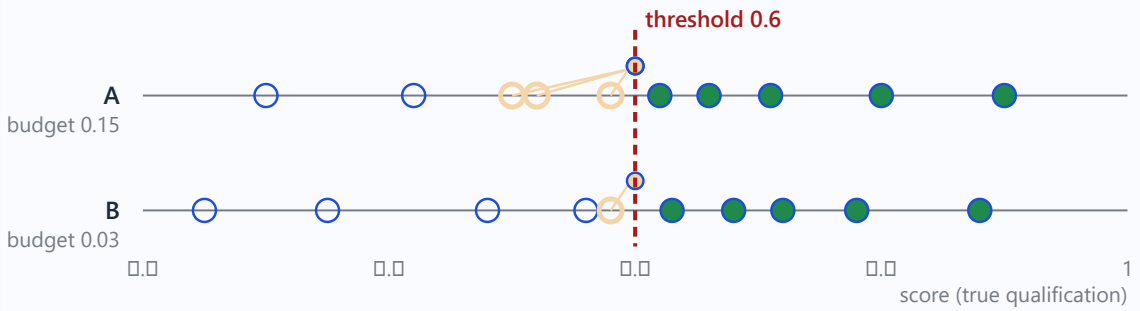


Figure 11.16: Gaming at threshold 0.6. Filled circles are qualified people (score ≥ 0.6). Three unqualified people from A and one from B get through by moving their scores to the threshold (orange rings and arrows).

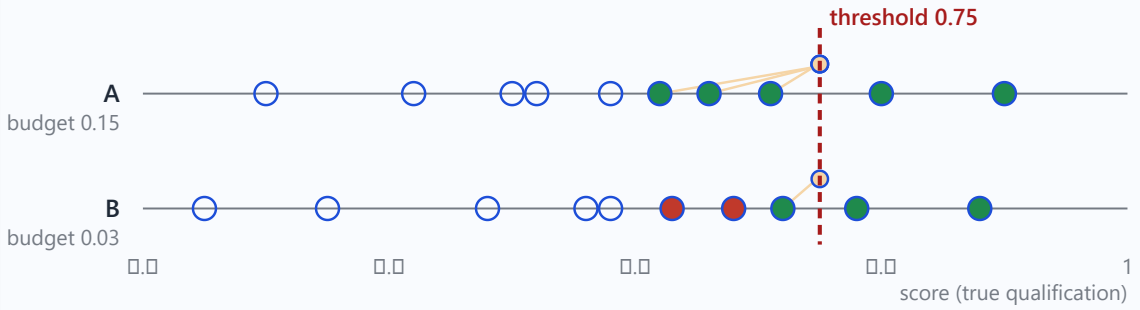


Figure 11.17: After the threshold is raised to 0.75. No unqualified person gets through. All five qualified members of A reach the threshold with their budget; the qualified members of B with scores 0.63 and 0.68 (red) cannot.

Figure 11.16 and Figure 11.17 compare the two stages. Raising the threshold is a crude way of imposing the cost of gaming on everyone. More refined solutions exist: giving weight to features that are hard to game, or making the resource needed for gaming (for example consulting or training) accessible to the low-budget group as well. A range computation can also be done in the example: for none of the unqualified to get through, the threshold must be greater than 0.73; for all of B's qualified to get through, the threshold must be at most *0.63 + 0.03 = 0.66*. Since the two conditions contradict each other, the two goals cannot be achieved together with a single threshold.

11.5.4 Algorithmic auditing

An **algorithmic audit** is the systematic examination of whether a decision system works in line with its purpose, correctly, and fairly. The audit covers not only the system's code but its whole life cycle.

What is audited?
: The data source, the target variable, performance by group, and human oversight are examined together. Whether the target variable is a proxy (the health spending example in 11.1.3) is often the most important question.

Documentation
: **Model cards** and dataset cards are short documents that make visible the assumptions, intended use, and known limitations of a model or dataset. A model card says, for example: "This model was trained on 2015–2020 credit applications; its accuracy is lower for applicants under 25."

After deployment
: The model continues to be monitored after it goes into use. **Data drift**, that is, the data the model encounters moving away over time from the training data, as well as objections and unexpected harms, are tracked regularly.

Participation
: The views of the affected groups are sought. A right to object and a clear chain of responsibility are defined.

The numerical part of the chapter gives the most concrete step of an audit: an **audit checklist**. In the list the base rates, the selection rate, TPR, FPR, PPV, and calibration are reported separately for each group. Because the impossibility theorems mean that not all of these criteria can be satisfied together, the reason for prioritizing a given criterion is justified in writing. Table 11.5 is the smallest form of such a report.

11.5.5 Can algorithms be neutral?

It is often said that algorithms are "neutral because they are mathematics". The findings of the chapter limit this claim in four ways:

- Algorithms are not independent of the values of the institutions that make the choices of target, data, criterion, and threshold. Even the answer to the question "what are we going to predict?" is a choice of values.

- Historical inequalities can be reflected in the training data and the target variables.
- Different fairness criteria produce conflicting results for the same decision (the impossibility theorems). Choosing one criterion means giving up another.
- After deployment, the model can create new feedback loops by changing behavior (11.5.6).

The basic difficulties are also clear. The notion of fairness changes with the context: the type of error that matters in a credit decision can be different in a medical screening decision. There are trade-offs among accuracy, privacy, and group fairness. Conceptions of fairness vary from culture to culture. A reasonable answer to these difficulties is to give up the claim of absolute neutrality. Instead, the effects of the decision are measured, the choices made are documented, and an objection mechanism is set up for the people affected.

11.5.6 Code trace: self-confirming data

The best-known example of a feedback loop is predictive policing. Patrols are sent to the region where incidents were frequent in past records. But an incident is recorded only where the patrol is present. In this case the records begin to reflect not the true distribution of incidents but where the patrol went (Ensign et al., 2018). The code below is the simplest model of this loop.

```
records = [6, 5]   # past records (region 0, 1)
actual = [3, 3]    # true daily incidents: equal!
for day in range(1, 5):
  target = 0 if records[0] >= records[1] else 1
  records[target] += actual[target]  # only there
  share = records[0] / sum(records)
  print(day, target, records, round(share, 2))
```

There are two regions in the code. The list `records` holds each region's past incident record: region 0 has 6 records and region 1 has 5. The list `actual` is the number of incidents that really occur in each region each day: 3 in both regions. That is, the two regions are in reality equal; a difference in the records is perhaps a coincidence. Each day the patrol goes to the region with more records (to region 0 in case of a tie). Only in the region the patrol goes to are that day's incidents entered in the records. `share` is region 0's share of the total records.

day	target	records	share	unseen incidents (region 1)
1	0 (6 ≥ 5)	[9, 5]	9/14 ≈ 0.64	3
2	0 (9 ≥ 5)	[12, 5]	12/17 ≈ 0.71	3
3	0 (12 ≥ 5)	[15, 5]	15/20 = 0.75	3
4	0 (15 ≥ 5)	[18, 5]	18/23 ≈ 0.78	3

Table 11.18: Trace of the feedback code. Start: records = [6, 5], share = 6/11 ≈ 0.55.

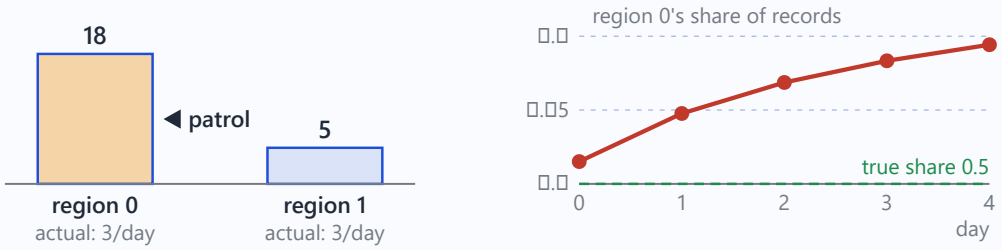


Figure 11.18: After four days the records are [18, 5]. The chart on the right shows region 0's share of the records: it rises from 0.55 to 0.78. The true share (green dashed line) stays fixed at 0.5.

Every day the patrol goes to region 0, because region 0's record is always larger. Region 0's record grows by 3 each day; the 3 incidents in region 1 are never seen and its record freezes at 5. Figure 11.18 shows the result: the share rises from 0.55 to 0.78. Because the data are collected by the model's decision, they "confirm" the model. An observer looking at the records may conclude "there is more than three times as much crime in region 0 as in region 1"; yet in reality the two regions are equal. The initial difference of a single record (6 against 5), perhaps merely a coincidence, grew itself into a lasting "truth".

Example 11.26

If the same code is run with `records = [5, 6]`, what is region 0's share after 4 days?

1. Day 1: region 1's record is larger (6 > 5); the patrol goes to region 1: [5, 9].
2. Day 2: [5, 12]. Day 3: [5, 15]. Day 4: [5, 18].
3. The share is $5/23 \approx 0.22$.

The patrol always goes to region 1. The initial difference of a single record this time grows in the opposite direction. What determines the direction of the result is not the true crime rates but the accidental initial difference.

There are ways of breaking the loop. The patrol can be sent to random regions from time to time; in this way some unbiased data are collected from every region. Or the records can be corrected by dividing by how much patrol was sent to the region. These two ideas correspond to the "exploration and exploitation" trade-off in learning theory: a system that only exploits the best-known option collects no information at all about the other options.

11.5.7 Real-world applications

The concepts of the chapter come up especially often in three areas:

Credit and banking

Scoring models. The real choice is between calibration and equal opportunity: the institution wants the score to convey the same risk in every group, while applicants want those who could repay to get the same chance.

Hiring and human resources

Résumé screening. Historical data bias (the Amazon example) and the four-fifths rule are the most frequent topics.

Healthcare

Diagnosis and treatment recommendations. The proxy target problem (spending ≠ need) can have grave consequences here.

The basic analysis applied in these areas is to study how changing a threshold affects the criteria. Since the chapter's example model gives only two scores, there are only three meaningful decision rules: reject everyone (threshold above 0.8), approve only those with 0.8 (threshold between 0.2 and 0.8), and approve everyone (threshold below 0.2).

Example 11.27

1. **Approve everyone.** In both groups the selection rate is 1, TPR is 1, and FPR is 1. Demographic parity and equalized odds hold. But PPV is the repayment rate of those approved (that is, of everyone): 0.50 in A and 0.32 in B. Predictive parity is broken. Accuracy falls to 0.50 in A and 0.32 in B.
2. **Approve only those with 0.8** (today's model). Predictive parity and calibration hold; demographic parity and equalized odds do not (Table 11.5). Accuracy is 0.80 in both groups.
3. **Reject everyone.** The selection rate, TPR, and FPR are 0 in both groups; demographic parity and equalized odds hold. PPV is undefined (nobody is approved). Accuracy is 0.50 in A and 0.68 in B. The institution gives no credit at all.

The two rules that satisfy equalized odds (approve everyone, reject everyone) render the model entirely useless. Rejecting everyone is the *TPR = 0* extreme case of the impossibility theorem in 11.3.1; approving everyone satisfies equalized odds while breaking predictive parity. The three rules show one thing concretely: unless the model's power to discriminate changes, all the criteria cannot be satisfied together just by playing with the threshold.

Common mistake

- **Counting the same rule applied to everyone as fairness:** In the strategic classification example the threshold is the same for the two groups, but the groups adapt to the rule at different costs. Formal equality does not bring equality of outcomes.
- **Taking records for reality:** In a feedback loop the records measure where the patrol went. Data collected by a process that is influenced by a decision need care when they are used to evaluate that decision.

- **Treating an audit as a one-off:** Because of data drift and strategic response, a model that is fair today may not be fair tomorrow. The audit continues after deployment.
- **Equating "human review" with accountability:** A human who merely approves the decision and does not question its rationale does not provide accountability. The person who objects must be able to reach the rationale of the decision.

11.6 Summary

- **Bias** comes from historical data, underrepresentation, proxy variables, and design and feedback loops. Removing the protected attribute from the model is not enough; proxies carry it. This is why outcomes are measured by group.
- The **confusion matrix** (TP, FP, FN, TN) is the source of all group criteria: selection rate $(TP + FP)/n$, TPR $TP/(TP + FN)$, FPR $FP/(FP + TN)$, PPV $TP/(TP + FP)$, base rate $(TP + FN)/n$.
- **Criteria:** demographic parity (selection rate; the four-fifths rule asks for a ratio ≥ 0.8), equalized odds (TPR and FPR; equal opportunity if only TPR), predictive parity (PPV), calibration (the realized rate of those with score s is s in every group). Equalized odds is the applicant's viewpoint and the last two are the institution's; demographic parity does not use the actual outcome at all.
- **Our example:** a model that is calibrated and has equal PPV (0.8); nevertheless the selection rates are 0.5/0.2, TPR 0.8/0.5, FPR 0.2/0.059. Accuracy is 80% in both groups; accuracy does not show on whom the errors fall.
- **Individual fairness** asks for similar scores for similar people: $|f(x) - f(y)| \le L \cdot d(x, y)$. Its difficulty is that the choice of the distance d is also a value judgment.
- **Incompatibility:** $FPR = p/(1 - p) \cdot (1 - PPV)/PPV \cdot TPR$ (Chouldechova). If base rates differ, predictive parity and equalized odds cannot hold together, except under perfect prediction. Kleinberg et al.: calibration and the two balances also hold together only under perfect prediction or equal base rates. In the COMPAS debate both sides were right by their own criterion.
- **Trade-off:** forcing equalized odds lowers PPV in B from 0.8 to 0.65; equal opportunity by randomization ($q = 0.6$) lowers accuracy from 0.8 to 0.512. A fairness constraint is added by pre-processing, in-processing, or post-processing; which criterion to choose is a value decision.
- **Fair division:** proportionality $v_i(X_i) \ge 1/n$, envy-freeness $v_i(X_i) \ge v_i(X_j)$. Cut and choose is proportional and envy-free for two people; cutting honestly is the maximin strategy, and the chooser is better off than the cutter. For two people proportionality ⇔ envy-freeness; for three or more, envy-freeness is stronger.
- **Algorithms:** the last diminisher and the moving knife are proportional for n people ($O(n^2)$ queries), Even–Paz with fewer queries; Selfridge–Conway is envy-free for 3 people, Aziz–Mackenzie for n people. A cut query is found by bisection; after k queries the error is at most $2^{-(k+1)}$.

- **Indivisible goods:** envy-freeness is not always possible; round-robin produces EF1, but greedy picking may not be a best response. The allocation that maximizes the Nash social welfare is Pareto optimal and EF1. In the example the price of fairness is $18/17 \approx 1.059$.
- **Ethics:** transparency, accountability, fairness, privacy; GDPR Article 22, the EU AI Act, KVKK. In an audit, the base rate, selection rate, TPR, FPR, PPV, and calibration are reported for each group, and the choice of criterion is justified.
- **Incentives:** people who respond to the algorithm set Goodhart's law in motion. In strategic classification, if the cost of gaming differs among groups, a new inequality arises; in a feedback loop, the data confirm the decision itself.

Concept	What it says	How it is computed
Demographic parity	Groups are approved at equal rates	Selection rates are compared
Equalized odds	Those in the same actual situation have equal chances	TPR and FPR are compared
Predictive parity	An approval carries the same meaning in the two groups	PPV is compared
Calibration	The score conveys the same risk in every group	Realized rate at each score level
Individual fairness	Similar scores for similar people	Lipschitz condition
Chouldechova identity	If base rates differ, the criteria conflict	$FPR = p/(1-p) \cdot (1-PPV)/PPV \cdot TPR$
Proportionality	Everyone receives ≥ 1/n in their own eyes	Cut and choose, last diminisher, moving knife
Envy-freeness	Nobody prefers another's share	Cut and choose (2), Selfridge–Conway (3)
EF1	Envy is at most one item	Round-robin and maximum Nash welfare (for non-negative additive values; in round-robin everyone takes the item most valuable in their own eyes)
Price of fairness	The welfare loss of a fairness constraint	Largest SW ÷ largest SW among fair allocations

Table 11.19: The concepts of the chapter at a glance.

11.7 Key terms

algorithmic fairness bias protected attribute proxy variable confusion matrix accuracy selection rate true positive rate (TPR) false positive rate (FPR) false negative rate (FNR) positive predictive value (PPV), precision negative predictive value (NPV) base rate group fairness demographic parity four-fifths rule equalized odds equal opportunity predictive parity

calibration within groups individual fairness Lipschitz condition impossibility theorems Chouldechova's identity balance for the positive/negative class loss function pre-processing / in-processing / post-processing fair division cake cutting valuation function Robertson–Webb query model cut query bisection proportionality envy-freeness cut and choose last diminisher moving knife indivisible goods bundle round-robin envy-free up to one good (EF1) utilitarian welfare egalitarian welfare Nash social welfare Pareto efficiency price of fairness transparency explainability accountability privacy strategic classification gaming Goodhart's law algorithmic audit model card data drift feedback loop

11.8 Test Yourself

1. The confusion matrix of a group D is TP = 24, FP = 6, FN = 16, TN = 54. What are the selection rate, TPR, FPR, PPV, and accuracy? Compared with group A of the chapter (selection 0.5; TPR 0.8; FPR 0.2; PPV 0.8), which criterion holds? What does the four-fifths rule say?

   Answer

   $n = 100$. Selection $(24 + 6)/100 = 0.30$; TPR $24/40 = 0.6$; FPR $6/60 = 0.1$; PPV $24/30 = 0.8$; accuracy $(24 + 54)/100 = 0.78$. Only predictive parity holds (0.8 = 0.8). The four-fifths rule: $0.30/0.50 = 0.6 < 0.8 \Rightarrow$ violated. Check with the identity: base rate $p = 40/100 = 0.4$; $(0.4/0.6) \cdot (0.2/0.8) \cdot 0.6 = 0.1$ ✓.

2. In one group the base rate is $p = 0.3$, $PPV = 0.6$, $TPR = 0.7$. What is the FPR? In a second group with base rate 0.5, if PPV and TPR are kept the same, what is the FPR? Can equalized odds hold in these two groups?

   Answer

   First group: $(0.3/0.7) \cdot (0.4/0.6) \cdot 0.7 = 0.3 \cdot (2/3) = 0.2$. Second group: $(0.5/0.5) \cdot (2/3) \cdot 0.7 = 7/15 \approx 0.467$. The TPRs are equal but the FPRs differ ($0.2 \neq 0.467$); equalized odds does not hold. When PPV is equal, the difference in base rates is carried into the FPR through the factor $p/(1 - p)$.

3. A model gives two scores: 0.7 and 0.1. In one group, of the 30 people with score 0.7, 21 repay the loan, and of the 70 people with score 0.1, 7 repay. Is the model calibrated in this group? What is

the base rate? What are the average scores of those who repay and of those who do not?

Answer

21/30 = 0.7 and *7/70 = 0.1*: at both score levels the realized rate equals the score, so the model is calibrated. The base rate is *(21 + 7)/100 = 0.28*; this equals the average score *(30 · 0.7 + 70 · 0.1)/100 = 0.28*. The average score of those who repay is *(21 · 0.7 + 7 · 0.1)/28 = 15.4/28 = 0.55*. Those who do not repay are 9 (with score 0.7) and 63 (with score 0.1); the average is *(9 · 0.7 + 63 · 0.1)/72 = 12.6/72 = 0.175*.

4. The cake is [0, 1]; the left half is vanilla and the right half chocolate. Ali values every part equally. For Ayşe all the vanilla is worth 2/5 and all the chocolate 3/5 (uniform within each half). (a) If Ali cuts and Ayşe chooses, what does each get in their own eyes? (b) If Ayşe cuts, where does she cut, and who gets what?

Answer

(a) Ali cuts at $x = 1/2$. For Ayşe the left is 2/5 and the right is 3/5; she chooses the right. Ali gets 1/2 and Ayşe 3/5. (b) Since all the vanilla is worth $2/5 < 1/2$ to Ayşe, the cut is in the chocolate. In the chocolate a unit of length is worth $(3/5)/(1/2) = 6/5$. $2/5 + (6/5)(x - 1/2) = 1/2 \Rightarrow x - 1/2 = 1/12 \Rightarrow x = 7/12$. Ali chooses the left piece and gets $7/12 \approx 0.583$; Ayşe gets 1/2. The chooser is again better off.

5. In the individual fairness condition, $L = 3$, $d(x, y) = 0.1$, and $f(x) = 0.25$. If scores lie in the interval [0, 1], in what range can $f(y)$ be?

Answer

$|f(y) - 0.25| \le 3 \cdot 0.1 = 0.3 \Rightarrow -0.05 \le f(y) \le 0.55$. Since a score cannot go below 0, the range is $[0, 0.55]$.

6. Two people share four indivisible items. Values: Ali (g1 8, g2 4, g3 2, g4 1), Ayşe (g1 5, g2 5, g3 4, g4 1). In round-robin with Ali starting, who takes what; is the allocation envy-free, is it EF1? Which allocation maximizes the Nash social welfare?

Answer

Ali g1 (8), Ayşe g2 (5), Ali g3 (2), Ayşe g4 (1). Ali {g1, g3} = 10, Ayşe {g2, g4} = 6. Ayşe values Ali's bundle at $5 + 4 = 9 > 6$: she envies. When g1 is removed, $4 \le 6$: EF1. The product is 60. Scanning the 16 allocations, the largest product is in the allocation Ali {g1, g4} = 9, Ayşe {g2, g3} = 9: 81. This allocation is envy-free (Ali values Ayşe's bundle at 6 and Ayşe values Ali's at 6) and its sum (18) is also the largest; in this example the price of fairness is 1.

7. The feedback code is run with `records = [7, 5]` and `actual = [2, 4]`; that is, more incidents really occur in region 1. After 4 days, what are the records and region 0's share of the records? What is the true share?

   Answer

   Every day region 0's record is larger and the patrol goes there: [9, 5], [11, 5], [13, 5], [15, 5]. The shares are 0.64, 0.69, 0.72, 0.75. The true share is $2/(2+4) \approx 0.33$. The records reverse the truth: most incidents are in region 1, but by the records region 0 looks three times as "dangerous".

11.9 Further reading

- S. Barocas, M. Hardt, and A. Narayanan, *Fairness and Machine Learning: Limitations and Opportunities* (2023): a textbook that presents group criteria, impossibility results, and causal approaches in detail and with examples; its online version is free.
- A. Chouldechova, "Fair prediction with disparate impact" (2017), and J. Kleinberg, S. Mullainathan, and M. Raghavan, "Inherent trade-offs in the fair determination of risk scores" (2017): the original papers of the two impossibility results in this chapter; the introductions of both can be read with little mathematics.
- A. D. Procaccia, "Cake cutting algorithms", *Handbook of Computational Social Choice*, Chapter 13 (2016): summarizes the Robertson–Webb model, the last diminisher, Even–Paz, and the complexity of envy-free protocols.
- S. J. Brams and A. D. Taylor, *Fair Division: From Cake-Cutting to Dispute Resolution* (1996): presents fair division methods with real dispute examples, in non-technical language.

11.10 References

1. J. Angwin, J. Larson, S. Mattu, L. Kirchner, “Machine Bias”, *ProPublica*, 2016.
2. H. Aziz, S. Mackenzie, “A discrete and bounded envy-free cake cutting protocol for any number of agents”, *IEEE FOCS*, 2016.
3. S. Barocas, M. Hardt, A. Narayanan, *Fairness and Machine Learning: Limitations and Opportunities*, MIT Press, 2023.
4. S. J. Brams, A. D. Taylor, *Fair Division: From Cake-Cutting to Dispute Resolution*, Cambridge University Press, 1996.
5. E. Budish, E. Cantillon, “The multi-unit assignment problem: theory and evidence from course allocation at Harvard”, *American Economic Review* 102(5), 2012.
6. J. Buolamwini, T. Gebru, “Gender Shades: Intersectional accuracy disparities in commercial gender classification”, *Proc. FAT**, 2018.

7. I. Caragiannis, C. Kaklamanis, P. Kanellopoulos, M. Kyropoulou, “The efficiency of fair division”, *Theory of Computing Systems* 50(4), 2012.

8. I. Caragiannis, D. Kurokawa, H. Moulin, A. D. Procaccia, N. Shah, J. Wang, “The unreasonable fairness of maximum Nash welfare”, *ACM EC*, 2016.

9. A. Chouldechova, “Fair prediction with disparate impact: a study of bias in recidivism prediction instruments”, *Big Data* 5(2), 2017.

10. J. Dastin, “Amazon scraps secret AI recruiting tool that showed bias against women”, *Reuters*, October 10, 2018.

11. L. E. Dubins, E. H. Spanier, “How to cut a cake fairly”, *American Mathematical Monthly* 68(1), 1961.

12. C. Dwork, M. Hardt, T. Pitassi, O. Reingold, R. Zemel, “Fairness through awareness”, *ITCS*, 2012.

13. D. Ensign, S. A. Friedler, S. Neville, C. Scheidegger, S. Venkatasubramanian, “Runaway feedback loops in predictive policing”, *Proc. FAT**, 2018.

14. S. Even, A. Paz, “A note on cake cutting”, *Discrete Applied Mathematics* 7(3), 1984.

15. M. Hardt, E. Price, N. Srebro, “Equality of opportunity in supervised learning”, *NeurIPS*, 2016.

16. L. Hu, N. Immorlica, J. W. Vaughan, “The disparate effects of strategic manipulation”, *Proc. FAT**, 2019.

17. A. R. Karlin, Y. Peres, *Game Theory, Alive*, AMS, 2017 (chapter on fair division).

18. J. Kleinberg, S. Mullainathan, M. Raghavan, “Inherent trade-offs in the fair determination of risk scores”, *ITCS*, 2017.

19. A. D. Procaccia, “Cake cutting algorithms”, in F. Brandt et al. (eds.), *Handbook of Computational Social Choice*, Chapter 13, Cambridge University Press, 2016.

20. European Parliament and Council, General Data Protection Regulation (GDPR), Regulation (EU) 2016/679, Art. 22 (applicable 2018).

21. European Parliament and Council, Artificial Intelligence Act, Regulation (EU) 2024/1689.

22. Law No. 6698 on the Protection of Personal Data (KVKK), *Official Gazette* of Turkey, April 7, 2016.

23. OECD, *Recommendation of the Council on Artificial Intelligence* (OECD AI Principles), 2019.

Behavioral and Collective Decisions

The players of classical game theory calculate flawlessly and care only about their own payoff. The people observed in the laboratory and in everyday life depart from this picture in systematic ways: they feel losses more heavily than gains, make small mistakes, and care about fairness and intentions. When they decide in groups they also deplete common resources, contribute too little to public goods, imitate one another and lose information, or, conversely, pool independent guesses and end up more accurate than any single member. This chapter works through, with numerical examples, the models that measure these deviations (prospect theory, quantal response, inequity aversion) and the models that explain group decisions (the commons, public goods and punishment, threshold projects, information cascades, the jury theorem).

LEARNING OUTCOMES

Behavioral decision making
Compute the value of a gamble with prospect theory; explain loss aversion, the endowment effect, and framing; show how quantal response changes the Nash prediction.

Social preferences
Explain the deviations observed in dictator, ultimatum, and trust games through inequity aversion and reciprocity; compute the Fehr–Schmidt acceptance threshold.

Common resources and public goods
Compare the equilibrium of the commons with the optimum; interpret Ostrom's principles, punishment and second-order free riding, and the equilibria of threshold projects.

Collective information
Compute the wisdom of crowds with the diversity prediction theorem, an information cascade with Bayes' rule, and the Condorcet jury theorem with the binomial distribution.

12.1 Behavioral decision making

In the classical model of game theory every player maximizes expected utility, makes no calculation errors, and cares only about their own payoff. These assumptions keep models simple and often yield surprisingly good predictions. Experiments with people, however, show that in some situations the predictions plainly fail. Behavioral game theory does not dismiss these deviations with "people are irrational"; it measures the direction and size of the deviations and builds them into the model. This section first treats individual decisions: choice under risk, the reference point, framing, and mistaken but sensible choice.

12.1.1 Starting with a game: the Public Goods Game

The game that ties together all the topics of this chapter is the **public goods game**. The rules are short. Four players receive 10 tokens each round. Each contributes as many as they like to a common pool. The tokens in the pool are multiplied by 1.6 and divided equally among the four players. Tokens not contributed stay in the player's pocket.

Let k_i be the number of tokens that player i contributes to the pool. The payoff in one round is written as:

$$u_i = 10 - k_i + 1.6 \cdot (k_1 + k_2 + k_3 + k_4) / 4 = 10 - k_i + 0.4 \cdot \Sigma k$$

The 0.4 in the formula is the multiplier (1.6) divided by the number of players (4). Figure 12.1 follows a single token. When a player puts 1 token into the pool, it becomes 1.6 and is divided by four; each player receives 0.4. The contributor gave up 1 token and got 0.4 back: a net of −0.6. Each of the other three players, without giving anything, gains 0.4; together they gain +1.2.

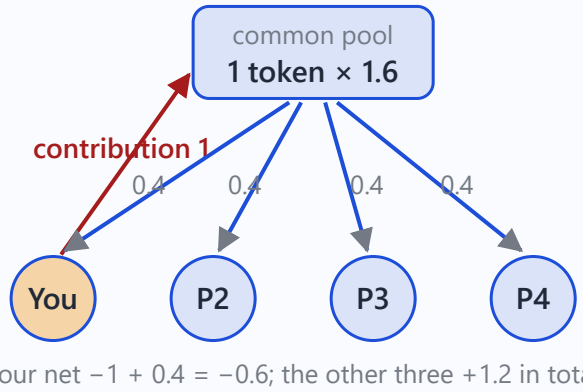


Figure 12.1: The path of one token in the Public Goods Game. A token put into the pool becomes 1.6 and is split 0.4 to each of the four players; the contributor's net is −0.6 and the other three players gain +1.2 in total.

This small calculation carries the central tension of the chapter. From the group's point of view every token contributed is valuable: 1 token given returns 1.6 to the group. From the individual's point of view every token contributed is a loss: 1 token given returns only 0.4 to the contributor. If everyone contributes 10, the pool is 40, grows to 64, and everyone receives 16. If nobody contributes, everyone keeps 10. The group optimum is full contribution, but the individual optimum (whatever the others do) is to contribute nothing. The same tension arises whenever an action serves the group's common interest but is costly to the individual: working on a joint project, keeping a park clean, contributing to open-source software.

Before starting the game one can ask two questions. First: in a group of four, how many tokens are contributed on average in the first round, and how many after ten rounds? Second: how do contributions change if anyone contributing fewer than 3 tokens is fined 2 tokens (the game's "Punished contribution" scenario)? The answer given by theory and the behavior observed in experiments diverge markedly. Both answers are worked out in 12.4. The sections in between build the models that explain the gap.

12.1.2 Bounded rationality

In classical decision theory the decision maker knows all the options, computes the consequences and probabilities of each, and picks the one with the highest expected utility. In 1955 Herbert Simon argued that this picture does not fit real people. Our information is incomplete, our time is limited, and the number of options we can hold in mind at once is small.

Definition 12.1

Bounded rationality (Simon, 1955): because the decision maker's information, time, and computational capacity are limited, they accept a "good enough" option that meets a given aspiration level instead of searching for the best of all options. This way of searching is called **satisficing**.

Consider a student looking for a summer internship. A fully rational decision maker would list all the companies, score each on salary, learning opportunities, and commute time, and pick the highest total. A real student applies to a few companies and accepts the first offer that meets the conditions "in Istanbul, at least minimum wage, a software job." This behavior lowers the cost of computation; the price is possibly missing a better offer.

The most important finding of behavioral research is this: people's deviations from the rational model are *systematic, not random*. The same errors are observed again and again, in different people and in different countries. Random errors cancel out on average and do not disturb models. Systematic errors, by contrast, always shift in the same direction; for exactly this reason they can be measured and added to a model with a formula. All the models in this chapter rest on this idea.

The causes of the deviations can be grouped under four headings: lack of information, time pressure, emotions, and shortcuts. The shortcuts are called **heuristics**: fast but sometimes misleading rules such as "if everyone is buying this product, it must be good" or "if it worked last time, it will work again." Four common systematic deviations are summarized in Table 12.1.

Deviation	Definition	Example
Overconfidence	Trusting one's own estimates more than warranted	Investors trading too much
Framing effect	The decision changes when the same option is presented differently	"90% success" and "10% failure" are perceived differently
Loss aversion	A loss feels stronger than a gain of equal size	Panic selling
Base rate neglect	Ignoring the prior probability when interpreting new evidence	Forgetting how common the disease is after a positive test

Table 12.1: Four systematic deviations and everyday examples.

Overconfidence is people's exaggeration of the accuracy of their own knowledge. Individual investors who trade often on the stock market earn less on average because of transaction costs, yet each believes their own forecast beats the market.

Base rate neglect is best understood with numbers. Suppose a disease affects 1% of the population. A test comes out positive for 90% of patients and, mistakenly, for 9% of healthy people. In a group of 1000 people there are 10 patients, and 9 of them test positive. About 89 of the 990 healthy people also test positive. Of the 98 people who test positive, only 9 are actually ill; that is, the probability of being ill after a positive result is about 9%. Most people think "the test is 90% accurate" and estimate this probability close to 90%: they ignore how common the disease is, that is, the prior probability. As this shows, the link Bayes' rule draws between prior and posterior probability can run quite against intuition. The same Bayesian calculation is used again for information cascades (12.6.2).

Framing and loss aversion, in turn, are explained by a separate model: prospect theory.

12.1.3 Prospect theory and the value function

In classical expected utility theory a person evaluates outcomes in terms of their *total wealth*. For someone whose wealth is 100,000 TL, winning 1,000 TL means wealth rising from 100,000 to 101,000. In 1979 Daniel Kahneman and Amos Tversky showed that this does not match how people actually evaluate. People perceive outcomes as gains or losses relative to a **reference point**. The reference point is often the status quo, but it can also be an expectation or a goal.

Definition 12.2

Prospect theory (Kahneman and Tversky, 1979) evaluates an option under risk as follows: each outcome x is written as a gain ($x \geq 0$) or a loss ($x < 0$) relative to the reference point, valued by a **value function** $v(x)$, weighted by probabilities, and summed. The value function used in these notes is:

$$v(x) = x^{0.88} \ (x \geq 0), \quad v(x) = -2.25 \cdot (-x)^{0.88} \ (x < 0)$$

The number $\lambda = 2.25$ here is called the **loss aversion** coefficient. The exponent 0.88 and $\lambda = 2.25$ are the values Tversky and Kahneman (1992) estimated from experiments.

The value function has three properties, and each explains a behavior. Figure 12.2 shows them on a single curve.

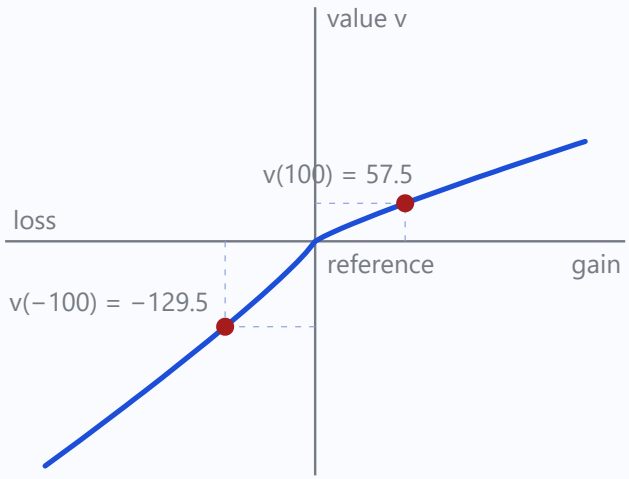


Figure 12.2: The value function of prospect theory. The curve passes through the reference point; it is concave on the gain side, convex on the loss side, and steeper on the loss side: $v(100) = 57.5$, $v(-100) = -129.5$.

1. Concave for gains. The function $x^{0.88}$ grows more slowly than x. Because the exponent is less than 1, the rate of increase falls as the number grows. In numbers: $v(100) = 100^{0.88} \approx 57.5$ and $v(200) = 200^{0.88} \approx 105.9$. When the amount doubles, the value does not double; it rises by only a factor of $2^{0.88} \approx 1.84$. The curve flattens as it goes right: the joy of the first 100 TL is greater than that of the second 100 TL. The consequence is risk aversion in the gain region: a sure 100 TL is valued more than a lottery paying 200 TL or 0 TL with equal probability, because $57.5 > \tfrac{1}{2} \cdot 105.9 \approx 53.0$.

2. Convex for losses. On the loss side the same diminishing sensitivity applies: the first 100 TL of loss hurts more than the second 100 TL of loss. But this time the result goes the other way. A sure loss of 100 TL has value $v(-100) \approx -129.5$; a gamble with a 200 TL loss or a 0 TL loss with equal

probability has value $\frac{1}{2} \cdot v(-200) \approx \frac{1}{2} \cdot (-238.3) \approx -119.1$. The risky option feels less bad: in the loss region people are risk seeking. This is the source of the behavior of "gambling rather than making the loss certain"; refusing to sell a losing stock because "it may come back" is one example.

3. Steeper for losses. A loss of the same size weighs $\lambda = 2.25$ times more than the gain: $v(-100) = -2.25 \cdot 57.5 \approx -129.5$. The curve has a kink at the reference point; approaching from the right the slope is gentle, from the left it is steep. This kink explains why even fair bets with zero expected value are rejected (12.1.4).

The effect of the reference point can be seen in a single example. An employee expecting a year-end bonus of 10,000 TL who receives 9,000 TL experiences it as a loss of 1,000 TL. For someone who expected no bonus at all, the same 9,000 TL is a large gain. The wealth of both has risen by the same amount; by classical theory both should be equally pleased. By prospect theory, satisfaction depends on where the outcome stands relative to the reference point.

Example 12.1

Let us compute the value function at a few points. On a calculator, $x^{0.88}$ is x raised to the power 0.88 (in Python, `x ** 0.88`).

1. $v(10) = 10^{0.88} \approx 7.6$. A gain of ten liras gives about 7.6 units of value.
2. $v(100) = 100^{0.88} \approx 57.5$. The amount grew tenfold; the value grew about 7.6-fold.
3. $v(1000) = 1000^{0.88} \approx 436.5$. Again a tenfold amount, again about a 7.6-fold value: for a power function the ratio is constant ($10^{0.88} \approx 7.6$).
4. $v(-100) = -2.25 \cdot 57.5 \approx -129.5$. A loss of the same size is 2.25 times heavier.

In this function, whenever the amount is multiplied by 10 the value grows by the same factor (about 7.6) at every scale. The ratio between loss and gain is likewise 2.25 at every scale.

12.1.4 Loss aversion: the coin-toss gamble

The cleanest test of loss aversion is a fair bet. A coin is tossed; heads wins 100 TL, tails loses 100 TL. The expected value is $\frac{1}{2} \cdot 100 + \frac{1}{2} \cdot (-100) = 0$. For a risk-neutral decision maker, taking the bet and declining it are the same. In experiments, however, the vast majority of people reject this bet. Prospect theory explains it as follows.

Example 12.2

1. **Outcomes relative to the reference.** Not betting is the reference point (0). The bet gives +100 with probability ½ and −100 with probability ½.
2. **Values.** $v(100) \approx 57.5$, $v(-100) \approx -129.5$.
3. **Prospect value.** $\frac{1}{2} \cdot 57.5 + \frac{1}{2} \cdot (-129.5) = 28.75 - 64.75 = -36.0$.

4. **Comparison.** The value of not betting is $v(0) = 0$. Since $-36.0 < 0$, the bet is rejected.

A bet with zero expected value is rejected because a loss weighs more than a gain.

How large must the gain be for the bet to be accepted? Let the gain be G and the loss again 100 TL. For the bet to be accepted, its prospect value must exceed zero. The boundary is the point where the two terms are equal.

Example 12.3

1. **Equation.** $\frac{1}{2} \cdot v(G) + \frac{1}{2} \cdot v(-100) = 0$, that is, $\frac{1}{2} \cdot G^{0.88} = \frac{1}{2} \cdot 2.25 \cdot 100^{0.88}$.
2. **Simplification.** The ½'s cancel; divide both sides by $100^{0.88}$: $G^{0.88} / 100^{0.88} = (G/100)^{0.88} = 2.25$.
3. **Removing the exponent.** Raise both sides to the power $1/0.88$: $G/100 = 2.25^{1/0.88} \approx 2.51$.
4. **Result.** $G \approx 100 \cdot 2.51 \approx 251$ TL.

To accept the risk of losing 100 TL, one demands a chance of winning about 251 TL in return. This ratio is independent of the amount: to accept a loss of 200 TL, about 503 TL is needed. Experiments likewise find that people typically demand more than twice the loss.

The operation in the third step is inverting the power equation. If $y^a = b$ then $y = b^{1/a}$, because $(b^{1/a})^a = b$. Here $a = 0.88$, $1/a \approx 1.136$, and $2.25^{1.136} \approx 2.51$. If the value function were linear (exponent 1), the threshold would be exactly $\lambda \cdot 100 = 225$ TL; because concavity makes large gains slightly less valuable, the threshold rises to 251.

12.1.5 Probability weighting

The second component of prospect theory is how probabilities are perceived. In classical theory a probability of 1% receives exactly weight 0.01. Experiments show that people feel small probabilities as larger than they are and large probabilities as smaller than they are. This perceived weight is called **probability weighting** and is written $w(p)$.

Probability p	0.01	0.50	0.99
Perceived weight w(p)	0.055	0.421	0.912

Table 12.2: Probability weighting (Tversky and Kahneman, 1992 estimate).

According to Table 12.2, a probability of 1% feels like about 5.5%: it is overstated more than fivefold. 50% is perceived as about 42% and 99% as about 91%: medium and large probabilities are underestimated. Certainty (p = 1), however, receives full weight; this is why moving from 99% to 100% is valued far more than moving from 50% to 51%.

These weights explain why two seemingly opposite behaviors are seen in the same person. Someone who buys a lottery ticket overstates a very small chance of winning: a 0.01% chance feels bigger to them. The same person, when insuring a home, also overstates a small probability of disaster and is willing to pay a premium above the expected damage. A model based on expected value cannot explain both behaviors together: according to it, a person either seeks or avoids risk. With probability weighting both come from the same source: the overstatement of small probabilities.

> **Common mistake**
>
> **"*w(p)* is a probability."** No, it is a decision weight. Probabilities sum to 1; weights do not: $w(0.01) + w(0.99) = 0.055 + 0.912 = 0.967 \neq 1$. Also $w(0.5) = 0.421$, not 0.5. Weights are not manipulated like probabilities; they only measure how the decision feels.

12.1.6 Code trace: prospect()

Let us trace with a short Python program how the value of a gamble changes as the gain amount changes. The program first defines the value function `v(x)`, then the prospect value of a gamble that gives +G or −K with equal probability, and tries four different gain amounts.

```
def v(x):
    if x >= 0: return x ** 0.88
    return -2.25 * (-x) ** 0.88
def prospect(G, K):      # ½: +G, ½: −K
    return 0.5 * v(G) + 0.5 * v(-K)
for G in [100, 200, 251, 300]:
    print(G, round(prospect(G, 100), 1))
```

The code runs line by line as follows:

- `v(x)` is the value function of Definition 12.2. If `x >= 0` the gain branch `x ** 0.88` runs; otherwise the loss branch `-2.25 * (-x) ** 0.88` runs. In the loss branch `-x` is a positive number; the power is taken and then multiplied by −2.25.
- `prospect(G, K)` is the value of the gamble that gives +G with probability ½ and −K with probability ½. Probability weighting is not used here; ½ is taken directly as the weight.
- The loop tries `G` = 100, 200, 251, and 300; the loss is always 100. For each, the value is rounded to one decimal and printed.

G	v(G)	v(−100)	Prospect value	Expected value	Decision
100	57.5	−129.5	−36.0	0	reject
200	105.9	−129.5	−11.8	+50	reject
251	129.3	−129.5	−0.1	+75.5	threshold

G	v(G)	v(−100)	Prospect value	Expected value	Decision
300	151.3	−129.5	+10.9	+100	accept

Table 12.3: Trace of `prospect(G, 100)`. The expected value is *(G − 100)/2*.

The program prints four lines: `100 -36.0`, `200 -11.8`, `251 -0.1`, `300 10.9`. Table 12.3 shows how each line comes about. At G = 100 the bet is fair, yet its prospect value is −36.0. At G = 200 the expected value is +50, so a risk-neutral person would take the bet at once; the prospect value is still −11.8 and the bet is rejected. At G = 251 the value is almost zero (−0.1): this is exactly the threshold of Example 12.3. At G = 300 the value is +10.9 and the bet is now accepted.

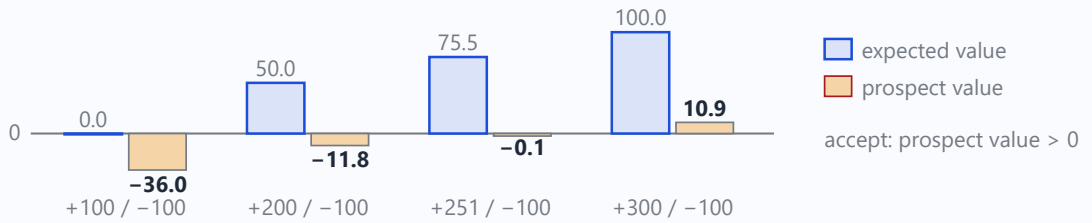


Figure 12.3: Expected value (blue) and prospect value (orange) for four bets. The expected value is zero or positive for every bet; the prospect value turns positive only when the gain exceeds 251 TL.

Figure 12.3 shows the two measures side by side. The blue bars (expected value) grow steadily from left to right. The orange bars (prospect value) are below zero for the first three bets. The +200 / −100 bet is rejected even though its expected value is +50. Loss aversion is largely responsible for this gap: with λ = 2.25 a loss weighs so heavily that the bet is not accepted until the gain exceeds 251 TL.

12.1.7 The endowment effect and status quo bias

Loss aversion shows up not only in gambles but also in the value people place on things they own. By classical theory the value of a mug does not depend on who holds it. Kahneman, Knetsch, and Thaler (1990) tested this prediction with a simple experiment. Half the students in a class were handed a mug at random. The mug owners were asked the least they would accept to sell it; those without a mug were asked the most they would pay to buy one.

The result was clear: the lowest selling price the owners demanded was about twice the highest price the buyers were willing to pay. Because the mugs were handed out at random, the tastes of the two groups are the same on average. By classical theory about half the mugs should have changed hands; in reality far fewer trades occurred than expected.

Definition 12.3

Endowment effect: owning an object raises the value placed on it. The lowest selling price an owner will accept is systematically higher than the highest price a buyer is willing to pay for the same object.

Prospect theory explains this effect through the reference point. The reference point of a mug owner is the state "I have a mug"; giving the mug up is a loss. The reference point of someone without a mug is "I have no mug"; getting the mug is a gain. Because a loss weighs λ times more, the seller asks for more than the buyer will pay.

Example 12.4

Let the mug's use value be 20 TL. To keep the calculation simple, take the value function to be linear (exponent 1) but keep the loss aversion coefficient $\lambda = 2.25$.

1. **Owner.** Giving up the mug is a loss of 20 TL from the reference point and feels like $-2.25 \cdot 20 = -45$. To make up for this loss the owner asks for at least 45 TL.
2. **Buyer.** Getting the mug is a gain of 20 TL: $+20$. The buyer pays at most 20 TL.
3. **Ratio.** $45 / 20 = 2.25 = \lambda$. The ratio of selling price to buying price is exactly the loss aversion coefficient.
4. **Result.** At no price between 20 and 45 TL can the two sides agree: the buyer will not pay more than 20 and the seller will not accept less than 45. No trade takes place.

The ratio measured in the experiment (about 2) is close to what this model predicts. An important consequence of the endowment effect is that resources do not flow automatically to those who value them most. The Coase theorem in economics says that if there are no transaction costs, who initially receives a resource does not affect efficiency. The endowment effect weakens this conclusion: the initial allocation determines how much trade takes place.

Its everyday counterpart is **status quo bias**. Departing from the default option is also perceived as a loss. When a retirement plan switches from "those who want to join must apply" to "everyone is enrolled automatically, and those who want out may leave," participation rises markedly. The options in the two rules are the same; only the reference point changes. This is also why default settings matter so much in software design: most users never change the default.

Common mistake

"The owner simply likes the mug more." No. The object is the same and the mugs were handed out at random; on average the owners' tastes do not differ from the buyers'. Only the reference point changes: the same mug is evaluated as a loss by the owner and as a gain by the

buyer.

12.1.8 The framing effect

The reference point can come from a person's own situation, but also from how the options are presented. Tversky and Kahneman (1981) showed this with the famous "Asian disease" problem. Participants were asked to choose between two programs against an epidemic expected to kill 600 people. One group was given the options in terms of lives saved, the other in terms of lives lost.

Frame	Sure program	Risky program	Majority choice
Gain	A: 200 people are saved	B: with probability 1/3 all 600 are saved, with probability 2/3 no one is saved	72% A (sure)
Loss	C: 400 people die	D: with probability 1/3 no one dies, with probability 2/3 all 600 die	78% D (risky)

Table 12.4: The two frames of the epidemic problem (Tversky and Kahneman, 1981).

The four programs in Table 12.4 are really two programs. In A, 200 people are saved, which means 400 die: A and C are the same outcome. In B, with probability 1/3 everyone is saved and with probability 2/3 everyone dies; D is the same: B and D are the same lottery. In all four programs the expected number saved is 200: $\tfrac{1}{3} \cdot 600 = 200$.

Definition 12.4

Framing effect: a change of preference when options with identical outcomes are presented differently (for example, as gains or as losses).

By expected utility theory the preference should not change, because the options in the two frames are identical. What is observed is that the majority switches from the sure option (72%) to the risky option (78%). Prospect theory explains this switch through the shape of the value function. In the gain frame the reference point is "everyone dies" and those saved are gains. In the gain region the value function is concave: 200 sure survivors feel worth more than a 1/3 chance of 600 survivors. In the loss frame the reference point is "no one dies" and the deaths are losses. In the loss region the function is convex: 400 sure deaths feel worse than a 2/3 chance of 600 deaths.

Example 12.5

This time the value function is applied to the number of people (without probability weighting).

1. **Gain frame, A.** $v(200) = 200^{0.88} \approx 105.9$.
2. **Gain frame, B.** $\tfrac{1}{3} \cdot v(600) \approx \tfrac{1}{3} \cdot 278.5 \approx 92.8$. $105.9 > 92.8$: the sure program A is chosen.
3. **Loss frame, C.** $v(-400) = -2.25 \cdot 400^{0.88} \approx -438.5$.

4. **Loss frame, D.** $\frac{2}{3} \cdot v(-600) \approx \frac{2}{3} \cdot (-626.5) \approx -417.7$. $-417.7 > -438.5$: the risky program D is chosen.

The same function produces two different choices in the two frames; the behavior of the majority is consistent with the model.

The framing effect has many practical counterparts. The sentence "90% of patients survive this operation" receives more approval than "10% die." The same price difference draws less reaction when presented as a "discount for paying cash" than as a "surcharge for paying by card." The lesson for an engineer designing a platform is clear: the way options are presented is not neutral; it changes user behavior.

12.1.9 Common mistakes in prospect theory

Because the concepts of prospect theory are close to one another, they are often confused. The four misconceptions below are the most common in exams and discussions.

Common mistake

- **"Loss aversion is risk aversion."** No. Risk aversion comes from the curvature of utility, loss aversion from the kink at the reference point (λ). The ±100 TL bet is rejected because of λ even if v is nearly linear for gains. Even with exponent 1 (no curvature) the prospect value is $\frac{1}{2} \cdot 100 - \frac{1}{2} \cdot 225 = -62.5$.
- **"$w(p)$ is a probability."** No, it is a decision weight: $w(0.01) + w(0.99) = 0.967 \neq 1$ and $w(0.5) = 0.421$.
- **"The reference point is always zero."** The reference can be an expectation, the status quo, or a goal. For someone expecting a 10,000 TL bonus, 9,000 TL is a loss.
- **"In the loss frame everyone chooses the risk."** The 78% is a majority tendency. The theory explains average behavior and does not predict individuals; 22% of participants chose the sure program in the loss frame too.

12.1.10 Quantal response

The Nash equilibrium assumes that every player always plays a best response. A best response is the strategy that maximizes the payoff while the others' choices are held fixed. Real people choose the better option more often, but not always. When the difference between options is small, they often choose the wrong one; as the difference grows, errors become rarer. A student choosing between two close answers on an exam may be wrong half the time; if one of the answers is obviously absurd, the student almost never picks it.

The most common way to model this behavior is **logit choice**. The idea is this: every option is given an "attraction" that increases with its payoff, and options are chosen with probability proportional to their attractions. The exponential function serves as the attraction. $e \approx 2.718$ is a fixed number, and e^x grows rapidly as x grows; it is always positive.

Definition 12.5

Logit choice: the player chooses each option a with a probability that depends on the option's expected payoff $u(a)$, namely

$$P(a) = e^{\lambda \cdot u(a)} / \Sigma_b \, e^{\lambda \cdot u(b)}$$

The denominator is the sum of the attractions of all options, so the probabilities add up to 1. The parameter $\lambda \geq 0$ measures how sharp the choice is: for $\lambda = 0$ the choice is completely random, and as $\lambda \to \infty$ it turns into the best response.

The λ here is unrelated to the $\lambda = 2.25$ of loss aversion; using the same letter in both models is merely a convention. In logit choice it helps to think of λ as "attention" or "sensitivity." For $\lambda = 0$ all exponents are $e^0 = 1$ and every option gets equal probability: the player pays no attention to payoffs at all. As λ grows, the effect of payoff differences grows; the attraction of the highest-payoff option overwhelms the others, and in the limit all the probability goes to it.

With two options the formula simplifies. Call the options "good" and "bad," and let the payoff difference be $d = u(good) - u(bad)$. Dividing numerator and denominator by $e^{\lambda \cdot u(good)}$ gives:

$$P(good) = 1 / (1 + e^{-\lambda \cdot d})$$

This is an S-shaped curve that depends only on the payoff difference. At difference 0 the probability is ½; as the difference grows it approaches 1.

Example 12.6

Two options with payoffs 10 and 8; the difference is $d = 2$. The probability of the good option is $P = 1 / (1 + e^{-2\lambda})$.

1. $\lambda = 0$: $1/(1 + 1) = 0.500$.
2. $\lambda = 0.5$: $e^{-1} \approx 0.368$, $1/1.368 \approx 0.731$.
3. $\lambda = 1$: $e^{-2} \approx 0.135$, $1/1.135 \approx 0.881$.
4. $\lambda = 2$: $e^{-4} \approx 0.018$, $1/1.018 \approx 0.982$.
5. $\lambda \to \infty$: $e^{-2\lambda} \to 0$, the probability tends to 1: best response.

λ	0	0.5	1	2	∞
P(payoff 10)	0.500	0.731	0.881	0.982	1

Table 12.5: Two options with payoffs 10 and 8 (difference 2): $P(\text{payoff } 10) = 1/(1 + e^{-2\lambda})$.

With three options the formula is the same. Let the payoffs be 4, 3, and 1 and $\lambda = 1$. The attractions are $e^4 \approx 54.6$, $e^3 \approx 20.1$, and $e^1 \approx 2.7$; the total is about 77.4. The probabilities are about 0.705, 0.259, and 0.035. The best option is chosen most often; the second option, because its payoff is close, still gets a probability of about a quarter; the clearly bad third option is almost never chosen. Logit choice turns the intuition "mistakes happen, but big mistakes are rare" into numbers in just this way.

In games this idea becomes an equilibrium concept. Each player's expected payoff depends on the others' probabilities; if the others also respond with logit choice, the probabilities are tied to one another.

Definition 12.6

Quantal response equilibrium (QRE; McKelvey and Palfrey, 1995): a probability profile in which every player responds with logit choice to the expected payoffs computed from the other players' equilibrium probabilities. In other words, plugging the others' probabilities into the logit formula returns each player's own probabilities. As $\lambda \to \infty$ the QRE tends to a Nash equilibrium.

QRE generalizes the Nash equilibrium: Nash is the extreme case of "zero error." In this way QRE can capture, with a single parameter (λ), the systematic deviations observed in experiments that Nash cannot explain. The most striking example is the asymmetric Matching Pennies game (12.1.11).

This is not the only way to model erroneous behavior. A separate model, **level-k reasoning**, limits not the error but the depth of reasoning (Nagel, 1995). A level-0 player plays randomly. A level-1 player assumes the others are level-0 and best-responds to them. In general, a level-k player takes the others to be level k−1. In the classic "guess 2/3 of the average" game, level-0 guesses an average of 50; level-1 says $\frac{2}{3} \cdot 50 \approx 33$; level-2 says $\frac{2}{3} \cdot 33 \approx 22$. The Nash equilibrium (everyone says 0) is reached only at infinite depth; in experiments the guesses mostly cluster around 33 and 22. QRE assumes "a little error at every step," level-k assumes "a limited number of steps"; the two models explain different experiments with different success.

12.1.11 Asymmetric Matching Pennies

The classical Matching Pennies game has no pure equilibrium; its only equilibrium is mixed, and both players play their options with probability ½. Goeree and Holt (2001) studied an asymmetric version of this game: the row player's payoff in one cell was made much larger. In Table 12.6 each cell lists the row player's payoff first, then the column player's.

Row \ Column	Left	Right
Top	(320, 40)	(40, 80)
Bottom	(40, 80)	(80, 40)

Table 12.6: Asymmetric Matching Pennies. Payoffs (row, column) in each cell.

The game has no pure equilibrium. At (Top, Left) the column player deviates to Right (40 → 80); at (Top, Right) the row player deviates to Bottom (40 → 80); at (Bottom, Right) the column player deviates to Left (40 → 80); at (Bottom, Left) the row player deviates to Top (40 → 320). The cycle never stops at any cell. To find the mixed equilibrium we use the **indifference principle**: if a player mixes in a mixed equilibrium, every pure strategy they mix over must give the same expected payoff. Otherwise they would always play the better one. For this reason a player's probability is set so as to make the *other* player indifferent.

Example 12.7

Let $p = P(Top)$ and $q = P(Left)$.

1. **The p that makes the column player indifferent.** The column player's expected payoff from Left is $40p + 80(1 - p)$ and from Right is $80p + 40(1 - p)$. Setting them equal gives $80 - 40p = 40 + 40p$, so $p = 1/2$.
2. **The q that makes the row player indifferent.** The row player's expected payoff from Top is $320q + 40(1 - q) = 40 + 280q$ and from Bottom is $40q + 80(1 - q) = 80 - 40q$. Setting them equal gives $320q = 40$, so $q = 1/8$.
3. **Interpretation.** The row player's probability of Top is 1/2 and does not depend at all on the row player's own payoff of 320: if 120 or 1000 were written instead of 320, p = 1/2 would remain. Only the column player's probability q changes.

This result is odd at first sight. The row player has a very large prize in cell (Top, Left); intuition says Top should be played more often. By the Nash equilibrium, however, a player's own payoffs have no effect on their own probabilities. Experiments side with intuition: in Goeree and Holt's (2001) experiments the row players clearly shifted toward the large payoff and played Top far more often than half the time.

QRE captures this "own-payoff effect." The row player's expected-payoff difference between Top and Bottom is $(40 + 280q) - (80 - 40q) = 320q - 40$, and the column player's difference between Left and Right is $(80 - 40p) - (40 + 40p) = 40 - 80p$. In QRE the two logit equations must hold simultaneously:

$$p = 1 / (1 + e^{-\lambda(320q - 40)}), \quad q = 1 / (1 + e^{-\lambda(40 - 80p)})$$

These two equations have no closed-form solution; the values are found numerically by computer. For example, guess a q, compute p from it, and from p compute a new q. The guess is adjusted by bisection until the two q values agree. The results are in Table 12.7.

λ	p = P(Top)	q = P(Left)
0	0.50	0.50
0.02	0.83	0.37
0.1	0.71	0.15
1	0.52	0.125
∞ (Nash)	0.50	0.125

Table 12.7: Logit QRE in asymmetric Matching Pennies.

Let us check one row of the table by hand. For $\lambda = 0.02$ and $q = 0.37$ the row player's difference is $320 \cdot 0.37 - 40 = 78.4$; $e^{-0.02 \cdot 78.4} = e^{-1.568} \approx 0.208$ and $p = 1/1.208 \approx 0.83$. For $p = 0.83$ the column player's difference is $40 - 80 \cdot 0.83 = -26.4$; $e^{0.528} \approx 1.70$ and $q = 1/2.70 \approx 0.37$. The two values confirm each other.

The table shows three regimes. For $\lambda = 0$ both players play randomly. When λ is small (0.02) the row player, drawn to 320, plays Top often (0.83); the column player senses this and reduces Left somewhat (0.37), but does not respond fully. As λ grows, the row player takes into account that the column player is escaping to Right; the probability of Top falls to 0.71, then to 0.52, and in the limit returns to Nash's 1/2. Behavior in experiments resembles intermediate values of λ. The strength of QRE is that it explains the "response to one's own payoff" that Nash ignores entirely, with no additional assumption, only a noise term.

Common mistake

"The row player's large payoff raises the row player's probability of Top in the mixed Nash equilibrium." No. In a mixed Nash equilibrium a player's probability is determined by the *opponent's* payoffs, because the aim is to make the opponent indifferent. When 320 changes, only q changes (1/8). People, however, also respond to their own payoff; this effect is absent in Nash and present in QRE.

12.2 Social preferences

The deviations in the previous section concerned how a person values their own payoff. The deviations in this section concern a person caring about *other people's* payoffs. In the classical model a player cares only about their own money; what the counterpart receives, how they were treated, or whether the outcome is fair does not enter the utility function. Laboratory experiments show that

this assumption, too, fails systematically. Social preference models write motives such as fairness, envy, guilt, and reciprocity into the utility function; the players are then still rational, but they maximize something different.

12.2.1 Dictator and ultimatum games

The simplest measures of social preferences are two small games.

Definition 12.7

Dictator game: one player (the dictator) divides 10 tokens between themselves and the counterpart; the counterpart can do nothing, and the division is carried out as stated.

Ultimatum game: the proposer offers s of the 10 tokens to the responder. If the responder accepts, the division $(10 - s, s)$ is carried out; if the responder rejects, both get 0.

For someone who cares only about money, the solution of both games is short. In the dictator game giving yields nothing, so the dictator gives 0. In the ultimatum game the responder should accept every positive offer, since even 1 token is better than 0. Knowing this, the proposer makes the smallest positive offer (1 token). This result is the subgame perfect equilibrium found by backward induction.

Observed behavior departs markedly from this prediction in both games. In a meta-analysis combining more than a hundred dictator game studies, dictators give on average about 28% of the pie, and a sizable fraction give nothing at all (Engel, 2011). In the ultimatum game offers are typically 40–50%, and offers below 20% are frequently rejected. Figure 12.4 compares theory and observation.

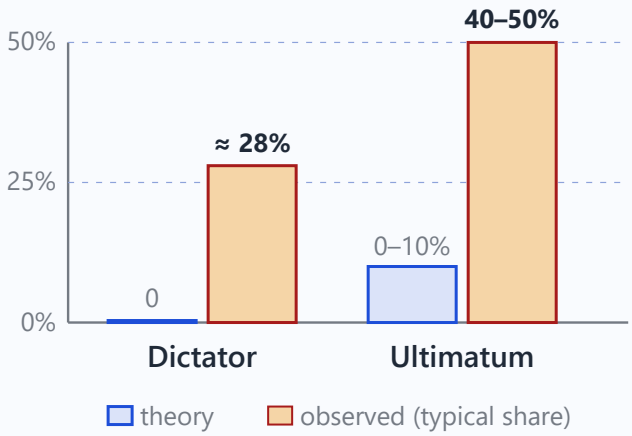


Figure 12.4: The share given to the counterpart in the dictator and ultimatum games. Theory (blue) predicts 0 in the dictator game and the smallest positive offer in the ultimatum game; the typical observed shares (orange) are about 28% in the dictator game and 40–50% in the ultimatum game.

Read together, the two games let us separate the motives. In the dictator game the other side has no power; the share given can come only from a sense of fairness or from altruism (caring about another's well-being). In the ultimatum game the proposer fears rejection; the share added on top of the dictator game's 28% is largely strategic, that is, meant to avoid rejection. The responder's rejection of low offers means giving up their own money: a responder who rejects 2 tokens loses 2 tokens but deprives the proposer of 8. This rejection behavior cannot be explained by the model that cares only about money. Social preference models explain both giving and rejection by writing these motives into the utility function.

12.2.2 Inequity aversion: the Fehr–Schmidt model

The model of Ernst Fehr and Klaus Schmidt (1999) assumes that people care not only about their own payoffs but also about the differences between payoffs. The difference is unpleasant in two directions: getting less than the counterpart arouses envy, getting more arouses guilt.

Definition 12.8

Inequity aversion (Fehr and Schmidt, 1999). In a two-person situation, if player i's money is x_i and the counterpart's is x_j, the utility is

$$u_i = x_i - \alpha \cdot \max(x_j - x_i, 0) - \beta \cdot \max(x_i - x_j, 0), \quad 0 \le \beta \le \alpha, \beta < 1$$

where α measures the pain of falling behind (envy) and β the discomfort of being ahead (guilt).

The $\max(\cdot, 0)$ expressions in the formula ensure that at most one of the two penalty terms is active. If the player is behind ($x_j > x_i$) only the α term is nonzero; if ahead, only the β term; at equality both are zero. Each of the constraints on the parameters has a meaning:

- $\beta \le \alpha$: people are more bothered by being behind than by being ahead.
- $\beta < 1$: someone who is ahead does not burn their own money merely to reduce inequality. Burning 1 token reduces the gap by 1 and gains at most $\beta < 1$ in utility; the loss of the burned token is larger.
- $\alpha = \beta = 0$: this gives the classical player who cares only about money. The model contains the classical model as a special case.

The model's prediction differs across the three games; let us solve each separately.

Example 12.8

Ultimatum game, responder. Let the offer be $s < 5$; the responder gets s, the proposer gets $10 - s$, and the responder is behind.

1. **Utility of accepting.** The difference is $(10 - s) - s = 10 - 2s$. $u = s - \alpha(10 - 2s)$.
2. **Utility of rejecting.** Both get 0, there is no difference: $u = 0$.
3. **Acceptance condition.** $s - \alpha(10 - 2s) \geq 0 \Leftrightarrow s + 2\alpha s \geq 10\alpha \Leftrightarrow s \geq 10\alpha / (1 + 2\alpha)$.
4. **Numerical values.** For $\alpha = 1$, $s \geq 10/3 \approx 3.33$; for $\alpha = 2$, $s \geq 20/5 = 4$; for $\alpha = 0$, $s \geq 0$ (every offer is accepted).

As α grows the threshold rises, but it never exceeds 5: $10\alpha/(1 + 2\alpha) < 5$ holds for every α. An equal split is acceptable to every responder.

Example 12.9

Dictator. If the dictator gives $s \leq 5$, the dictator stays ahead (or equal); only the β term is active.

1. **Utility.** The dictator's money is $10 - s$, the difference is $10 - 2s$: $u = (10 - s) - \beta(10 - 2s)$.
2. **Rearranging.** $u = 10 - 10\beta - s + 2\beta s = 10 - 10\beta + s(2\beta - 1)$.
3. **Slope.** u is a linear function of s with slope $2\beta - 1$. If $\beta > 1/2$ the slope is positive; the dictator gives as much as possible, that is, 5 (splits equally). If $\beta < 1/2$ the slope is negative; the dictator gives nothing. If $\beta = 1/2$ the slope is 0 and every share from 0 to 5 gives the same utility.
4. **Giving more than 5.** If $s > 5$ is given, the dictator falls behind and the α term comes into play. Money falls and the pain of envy begins; for this reason the dictator does not give more than 5.

For example, a dictator with $\beta = 0.6$ gives 5 and gets $u = 5$; one with $\beta = 0.3$ gives 0 and gets $u = 10 - 3 = 7$.

This result explains the two clusters in experiments: some dictators give nothing, some split equally. In the model, intermediate values (for example giving 2 or 3) arise only in the boundary case $\beta = 1/2$; the intermediate values seen in real data suggest that other motives (for example the wish to look good) are also at work.

Public goods. A player who is sufficiently inequity averse contributes if the others contribute, because by not contributing they would stay ahead of the others and the β term would bother them. If the others do not contribute, neither does this player, because contributing alone would leave them behind and the α term would bother them. This behavior is called **conditional cooperation**, and such a player is a **conditional cooperator**. The dynamics of laboratory public goods experiments (12.4) come largely from the behavior of players of this type.

12.2.3 The ultimatum acceptance threshold, step by step

For a responder with $\alpha = 1$ the acceptance threshold is 10/3. To see which offers are accepted when offers are whole numbers, let us check the offers one by one. The utility of accepting is $u = s - 1 \cdot (10 - 2s) = 3s - 10$ and the utility of rejecting is 0.

Offer s	Division (proposer, responder)	Difference 10 − 2s	Utility of accepting u	Decision
1	(9, 1)	8	1 − 8 = −7	reject
2	(8, 2)	6	2 − 6 = −4	reject
3	(7, 3)	4	3 − 4 = −1	reject (narrowly)
4	(6, 4)	2	4 − 2 = +2	accept
5	(5, 5)	0	5	accept

Table 12.8: Response of a responder with $\alpha = 1$ to whole-number offers.

Table 12.8 makes the same comparison in every row. An offer of 1 token does not make up for the pain of a gap of 8 tokens; the utility is −7 and rejecting (0) is better. When the offer rises by 1, the responder's money rises by 1 and the gap falls by 2; the utility rises by 3 in total. So the utility climbs in steps of three, −7, −4, −1, +2, +5, and crosses zero between 3 and 4. Figure 12.5 shows the same numbers as bars.

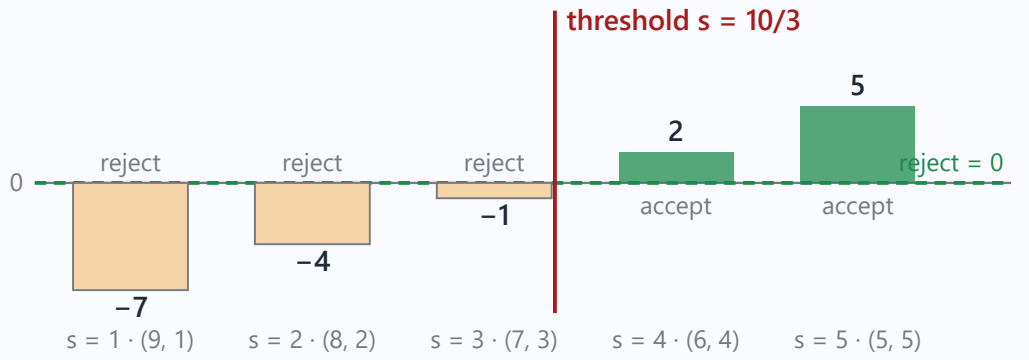


Figure 12.5: Acceptance utility of a responder with $\alpha = 1$. The bars are below zero for $s = 1, 2, 3$ (reject) and above it for $s = 4, 5$ (accept); the threshold is $s = 10/3$.

Since the threshold is $10\alpha/(1 + 2\alpha) = 10/3 \approx 3.33$, the smallest whole-number offer that is accepted is 4. If the proposer knows the responder's α, the proposer offers 4 and gets 6; had the proposer offered 3, it would be rejected and the proposer would get 0. This holds even if the proposer cares only about money. Rejecting a low offer is not a mistake but the result of a sense of fairness; the proposer, too, is a rational player who takes this into account. The model thus explains, within one framework, both the responder's rejection and the proposer's seemingly "generous" offer.

Common mistake

"If the threshold is 3.33, then 3 tokens are accepted." No. The acceptance condition is $s \geq 3.33$; 3 does not satisfy it ($u = -1$). When the threshold is not a whole number, the smallest accepted offer is the threshold rounded up. Moreover the threshold formula holds only for $s < 5$; for offers $s > 5$ the responder is ahead and the β term is active.

12.2.4 Reciprocity and the trust game

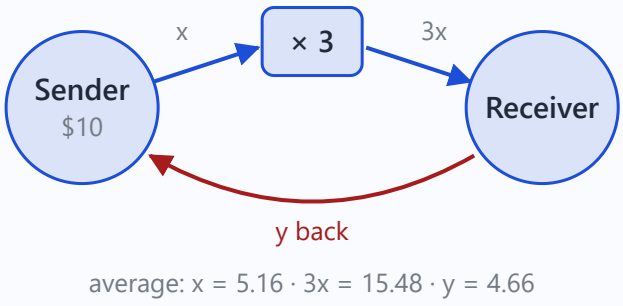
Inequity aversion looks only at the distribution of outcomes: who got how much? People, however, often react not to the outcome but to the *intention* that led to it. The classic experiment measuring this is the trust game.

Definition 12.9

Trust game (Berg, Dickhaut, and McCabe, 1995): the sender sends the receiver any amount x of 10 dollars. The amount sent is tripled; the receiver gets $3x$ and sends back any amount y of it to the sender. The sender earns $10 - x + y$ and the receiver $3x - y$.

The design of the game is chosen with care. Because every dollar sent is tripled, with full trust and full reciprocity the two sides together earn the most: 10 dollars become 30. But the receiver has no obligation at all to send anything back. The experiment was played one-shot and under full anonymity: the participants did not know one another, would never meet again, and even the experimenters did not know who did what individually. Strategic reasons such as reputation or future reward were thus eliminated.

For players who care only about money the solution is found by backward induction. The last move is the receiver's; sending back only reduces their own payoff, so $y = 0$. The sender, foreseeing this, knows that anything sent will be lost; so $x = 0$. The theory's prediction: nothing is sent, and both sides stay where they started.



average: x = 5.16 · 3x = 15.48 · y = 4.66

Figure 12.6: The trust game. The sender sends x, the amount is tripled, and the receiver sends back y. In the experiment the averages were $x = 5.16$ dollars, $3x = 15.48$ dollars, and $y = 4.66$ dollars.

In the experiment, however, senders sent 5.16 dollars on average and receivers sent back 4.66 dollars on average (Figure 12.6). The great majority of senders sent something; some receivers sent back more than was sent, some sent nothing. The average sender started with 10 dollars and ended with $10 - 5.16 + 4.66 = 9.50$ dollars; the average receiver kept $15.48 - 4.66 = 10.82$ dollars.

Definition 12.10

Reciprocity (Rabin, 1993): a person's responding to the counterpart's *intention*, answering kind behavior with kindness and unkind behavior with punishment.

The difference between reciprocity and inequity aversion can be seen in a thought experiment. Suppose a responder in the ultimatum game receives an offer of 2 tokens. In the first case the other person chose the offer themselves. In the second case a lottery that neither side could influence determined the offer. Inequity aversion cannot tell the two cases apart, because in both the outcome is (8, 2) and the utility is the same. Reciprocity can: in the first case the other person has a selfish intention and rejection punishes that intention; in the second there is no intention to punish and acceptance is more likely. The receiver's sending something back in the trust game is also explained by reciprocity: the sender has shown trust, and the receiver responds to this good intention.

Common mistake

"Trusting paid off." Not on average. What came back (4.66 dollars) was less than what was sent (5.16 dollars); the average sender lost 0.50 dollars. Even so, most people sent. The surprising result of the experiment is not that trust paid but that it was so widespread although it did not pay. In terms of the total, trust did create value: because each dollar sent was tripled, the two sides' combined earnings rose from 10 dollars to an average of $10 + 2 \cdot 5.16 \approx 20.3$ dollars.

12.3 Collective decisions and common resources

So far the decisions of one or two people have been examined. From here on the focus is the outcome a group produces together. The basic problem with group outcomes is that each individual, when making their own decision, does not account for its effect on the other members. This section models the problem in its most classic form, the overuse of a common resource; it then treats remedies and the welfare criteria used to evaluate outcomes.

12.3.1 Individual and collective interest

A decision made by a group is called a **collective decision**. In group decisions social influence, pressure to conform, and common interests play a role. Communities make decisions by majority vote, by **consensus** (a common decision that everyone agrees to), or by more complex voting

systems. But a group's outcome is not always determined by an explicit vote. Often the outcome is the sum of small decisions each individual makes alone: nobody votes for "let there be a traffic jam," but everyone choosing their own route produces one.

Individual interest maximizes a person's own utility, **collective interest** the group's total utility. The two often conflict. Three examples show different forms of this conflict:

- **Traffic.** Every driver chooses the shortest route for themselves; the total delay is greater than central planning would give. This is the basic example in the topic of selfish routing and the price of anarchy: in the two-link Pigou network the selfish equilibrium raises the total delay by a factor of 4/3 over the optimum.
- **Work groups.** Everyone focuses on their own task; a joint project that nobody owns alone stalls. This is a public goods problem (12.4).
- **Common resources.** Fisheries, pastures, groundwater, the atmosphere: resources that everyone can use but that shrink as they are used are overused.

The classic name for the last example is the **tragedy of the commons**. In an article published in 1968, Garrett Hardin gave the example of herders letting animals graze on a pasture open to all. For each herder, adding one more animal brings the whole gain to themselves, while the cost of the pasture's degradation is divided among all the herders. So every herder keeps adding animals and the pasture is exhausted. Hardin's verbal argument can be turned into a numerical model with game theory.

12.3.2 The tragedy of the commons: a numerical model

Four herders graze animals on a common pasture. Let the number of animals herder i grazes be g_i and the total number of animals $G = g_1 + g_2 + g_3 + g_4$. As the pasture becomes crowded each animal gets less feed; let the value per animal be $100 - G$. At 100 animals the pasture is completely exhausted and the animals become worthless. A herder's payoff is the number of animals times the value per animal:

$$u_i = g_i \cdot (100 - G)$$

This model closely resembles the Cournot model: there firms choose a quantity and the price falls with total output; here herders choose a number of animals and the value per animal falls with the total. So the solution method is the same: best responses are found with a derivative. First the group optimum is computed, then the individual optimum.

Example 12.10

Social optimum. The group's total payoff is $\Sigma u_i = G \cdot (100 - G)$; which herder has how many animals does not change the total.

1. **Expansion.** $G(100 - G) = 100G - G^2$.
2. **Derivative.** The derivative of $100G$ is 100 and that of G^2 is $2G$: the derivative is $100 - 2G$.

3. **Set to zero.** $100 - 2G = 0 \Rightarrow G = 50$. The second derivative is −2, so this is a peak.
4. **Value.** The total payoff is $50 \cdot 50 = 2{,}500$. With an equal split each herder grazes 12.5 animals and earns $12.5 \cdot 50 = 625$.

The optimum can be thought of as an agreement: "everyone grazes 12.5 animals." The question is whether this agreement is self-enforcing. For that we ask whether one herder would want to deviate while the others keep the agreement.

Example 12.11

A single herder deviates. The other three herders graze 12.5 each, 37.5 animals in total. If herder 1 grazes g animals, the total is $37.5 + g$.

1. **Payoff.** $u_1 = g \cdot (100 - 37.5 - g) = g \cdot (62.5 - g) = 62.5g - g^2$.
2. **Derivative.** $62.5 - 2g = 0 \Rightarrow g = 31.25$.
3. **The deviator's payoff.** The total number of animals is $37.5 + 31.25 = 68.75$, the value per animal is 31.25; the payoff is $31.25 \cdot 31.25 \approx 976.6$. This is far more than the 625 under the agreement.
4. **The others' payoff.** Each gets $12.5 \cdot 31.25 \approx 390.6$; it falls from 625 to 390.6.

The agreement is not self-enforcing: every herder wants to deviate. The deviator's gain comes from the others' loss.

If everyone acts on the same logic, the outcome is the Nash equilibrium. To find it, write herder i's general best response. Let the others' total number of animals be G_{-i}:

$$u_i = g_i(100 - g_i - G_{-i}) \;\Rightarrow\; \text{derivative } 100 - 2g_i - G_{-i} = 0 \;\Rightarrow\; g_i = (100 - G_{-i}) / 2$$

Example 12.12

Symmetric Nash equilibrium.

1. **Symmetry.** The four herders are in the same position; in equilibrium all choose the same g. Then $G_{-i} = 3g$.
2. **Equation.** $g = (100 - 3g)/2 \Rightarrow 2g = 100 - 3g \Rightarrow 5g = 100 \Rightarrow g^* = 20$.
3. **Total and payoff.** $G = 80$, the value per animal is 20; the payoff per herder is $20 \cdot 20 = 400$, 1,600 in total.
4. **Check.** When the other three graze 60 animals, the best response is $(100 - 60)/2 = 20$. Nobody wants to deviate ✓.

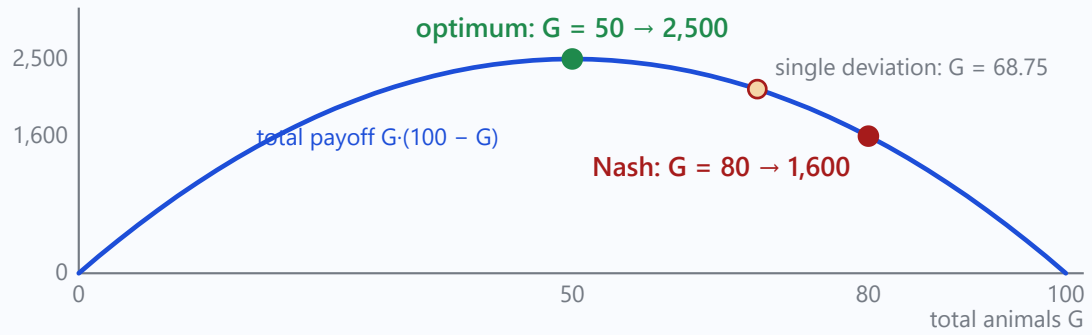


Figure 12.7: The total payoff curve $G(100 - G)$. The optimum is 2,500 at $G = 50$; when a single herder deviates it falls to $G = 68.75$, and when everyone best-responds (Nash) to $G = 80$ and 1,600.

Figure 12.7 shows the three situations on the same curve. The top of the curve is the optimum. A single herder's deviation pushes the total to the right of the peak; when everyone deviates the total number of animals rises to 80 and the group payoff falls to 1,600. The tragedy is this: in equilibrium the total payoff is only 64% of the optimum, and each herder gets 400 instead of the 625 they would have received had the agreement been kept. Everyone has calculated correctly from their own standpoint; the outcome is bad for everyone. This structure is a many-player, continuous-strategy form of the Prisoner's Dilemma.

Why the equilibrium is far from the optimum is explained by the concept of an **externality**. An externality is an effect of one person's decision on others that does not enter their own calculation. When a herder adds one more animal, the value per animal falls by 1, and this fall is passed on to every animal on the pasture. The herder feels only the part of the loss that falls on their own animals; in the symmetric case, being one of four herders, this share is roughly 1/4. The remaining 3/4 falls on the other herders and does not enter the herder's calculation. Because every herder ignores part of the cost, everyone grazes too many animals.

How is the equilibrium reached? If the herders best-respond in turn, the process converges to this equilibrium. Let the pasture start empty. Herder 1, with nobody else present, grazes $(100 - 0)/2 = 50$; herder 2, against 50, grazes 25; herder 3, against 75, grazes 12.5; herder 4, against 87.5, grazes 6.25. In later rounds herder 1, against the others' 43.75, reduces the herd while the other herders increase theirs. As the rounds go on all four numbers approach 20. In this sense the equilibrium is stable: it is not merely an assumption but the point a reasonable learning process arrives at.

12.3.3 More herders, and remedies

When the number of herders n changes, the same calculation can be done in general. In the symmetric equilibrium $g = (100 - (n - 1)g)/2$, that is, $(n + 1)g = 100$ and

$$g^* = 100 / (n + 1), \quad G^* = 100n / (n + 1)$$

n	g* (per herder)	G*	Total payoff	Ratio to optimum
1	50	50	2,500	100%
2	33.33	66.7	2,222	88.9%
4	20	80	1,600	64%
10	9.09	90.9	826	33.1%
100	0.99	99	98	3.9%

Table 12.9: Equilibrium by number of herders. The optimum is $G = 50$, with a total of 2,500, in every case.

Table 12.9 shows how the tragedy deepens with the number of herders. A single herder ($n = 1$) is the sole owner of the pasture; they feel the whole cost themselves and choose the optimum. With two herders the loss is small (the total is 88.9% of the optimum). With ten herders the group does not even get a third of the possible payoff. With a hundred herders the pasture is almost completely exhausted ($G^* = 99$) and the total payoff approaches zero: as $n \to \infty$, $G^* \to 100$ and the total $\to 0$. The reason is the growth of the externality: on a pasture with n herders each herder feels only roughly $1/n$ of the damage they cause. This is why common resources are exhausted more easily in large, anonymous groups.

Three kinds of remedy have been proposed for escaping the tragedy.

1. Pigouvian tax. In this remedy, named after the economist Arthur Pigou, a tax is levied on each unit that creates an externality. The cost the individual ignores is thereby added back to their calculation. If a herder pays a tax t per animal, the payoff becomes $u_i = g_i(100 - G - t)$.

Example 12.13

1. **Best response.** Derivative $100 - t - 2g_i - G_{-i} = 0 \Rightarrow g_i = (100 - t - G_{-i})/2$. The formula is the same as without the tax; only $100 - t$ replaces 100.
2. **Symmetric equilibrium (n = 4).** $g = (100 - t - 3g)/2 \Rightarrow g = (100 - t)/5$.
3. **Targeting the optimum.** We want $G = 4g = 50$, that is, $g = 12.5$: $(100 - t)/5 = 12.5 \Rightarrow 100 - t = 62.5 \Rightarrow t = 37.5$.
4. **Check.** With the tax each herder grazes 12.5 animals; a herder nets $12.5 \cdot (100 - 50 - 37.5) = 156.25$ and the total tax revenue is $4 \cdot 12.5 \cdot 37.5 = 1{,}875$. The herders' net payoffs plus the tax revenue sum to $4 \cdot 156.25 + 1{,}875 = 2{,}500$, that is, the optimum.

For general n the same calculation gives $g = (100 - t)/(n + 1)$ and, for the optimum, $t = 100 - 50(n + 1)/n$; for example, for $n = 9$, $t \approx 44.4$.

The numbers in the check show an important property of the Pigouvian tax: the tax does not destroy total value, it only moves it around. If the revenue is redistributed to the herders in equal shares, independent of the number of animals, each herder receives 625; everyone is better off than in the

no-tax equilibrium. The size of the tax is not arbitrary either: at the optimum, adding one more animal lowers the value of each of the other herders' 37.5 animals by 1. The tax equals this externality (37.5) at the optimum.

2. Quotas or permits. The total number of animals is limited directly to 50 and the rights are divided among the herders. If rights can be bought and sold, the herders who will use them most productively buy them. Fishing quotas and carbon emission permits are examples of this remedy. A quota fixes the quantity directly; a tax fixes the price and leaves the quantity to individuals' decisions.

3. Local institutions. The work of Elinor Ostrom showed that local communities can sustain their resources for centuries through monitoring, graduated sanctions, and shared rules. For this work Ostrom received the Nobel Prize in economics in 2009. The principles of this third path are given in 12.3.4.

12.3.4 Ostrom's design principles

According to Hardin there were only two ways out of the tragedy of the commons: either the state controls the resource centrally, or the resource is privatized and divided. Ostrom (1990) questioned this dichotomy. She studied many local governance arrangements that had lasted for centuries, from the Alpine pastures of Switzerland to the village forests of Japan, from the irrigation associations of Spain to the irrigation systems of the Philippines. In these arrangements the resource belonged neither to the state nor was it divided; the users governed themselves through shared rules. Ostrom identified the principles that the successful cases share.

Principle	Meaning
1. Clear boundaries	It is clear who may use which resource.
2. Rules suited to local conditions	Rules for use and contribution are tailored to the resource.
3. Participation in the rules	Those affected by the rules can change them.
4. Monitoring	There are monitors who are accountable to the users.
5. Graduated sanctions	A first violation draws a mild penalty, repetition a heavy one.
6. Cheap conflict resolution	Disputes are settled quickly and locally.
7. Right to organize	External authorities recognize local rules.
8. Nested enterprises	Large resources have layered institutions.

Table 12.10: Ostrom's eight design principles (Ostrom, 1990).

The principles can be read in the language of game theory. Clear boundaries fix the set of players; they keep outsiders from coming in and exhausting the resource, and make clear "who is playing against whom." Participation in the rules ensures that rules are made with local knowledge and are seen as legitimate. Monitoring makes it possible to notice deviation in a repeated game; if deviation goes unseen, no threat of punishment works. Graduated sanctions do two things: a small mistake or misunderstanding does not lead to a penalty that would break up the community, while repeated

violation becomes steadily more costly. Cheap conflict resolution settles disputes before they grow where the rules are unclear. All of these mechanisms turn the conditions that sustain cooperation in repeated games (observation, punishment, a long relationship) into concrete institutions.

One of the cases Ostrom studied is from Turkey: the inshore fishermen of Alanya. After rising competition and fights over fishing grounds in the 1970s, the fishermen set up their own arrangement. At the start of each season the fishermen with fishing rights are listed and named fishing spots are allocated by lot. During the season the boats rotate, moving each day to the next spot in line with the direction of the fish migration. In this way everyone gets an equal share of good and bad spots, the number of boats in disputes falls, and the rule is easy to monitor: the boat fishing in the wrong spot is noticed at once by that spot's right holder for the day. The rule did not come from the state, nor was the resource divided and sold; yet the institutional arrangement works.

Common mistake

"The solution for a common resource is either the state or the market." Ostrom's local institutions are a third way. A resource can be sustained, without being nationalized or privatized, through rules that the users set and monitor themselves. This does not mean that every common resource will manage itself: the principles are conditions found in the successful cases and missing in the failed ones.

12.3.5 Social welfare criteria and the veil of ignorance

In the common-pasture example the word "optimum" was used for the situation that maximizes the total payoff. But the total is not the only way to evaluate a group's outcomes. A policy can raise the total while leaving a few people very badly off. For this reason several criteria are used for **social welfare**.

Definition 12.11

Let the individuals' utilities be $u_1, \ldots, u_n$.

- **Utilitarian criterion**: $SW = \Sigma\, u_i$. The option with the largest sum is best.
- **Egalitarian criterion**: $min\ u_i$. The option that maximizes the utility of the worst-off individual is best.
- **Nash social welfare**: $\Pi\, u_i$. The option with the largest product of utilities is best.

The three criteria can rank the same options differently. The optimum on the common pasture ($G = 50$, total 2,500) is the best by the utilitarian criterion. Nash social welfare stands somewhere in between: the product cares about the total but heavily penalizes any utility approaching zero, because a single zero wipes out the whole product.

Which criterion to choose is a philosophical question, but decision theory provides a tool that sharpens it. In the famous thought experiment of the philosopher John Rawls, the rules of a society are chosen without knowing who will be in which position. The decision maker stands behind a **veil of ignorance**: they do not know whether they will be born rich or poor, healthy or ill. The social choice then becomes an individual decision problem under uncertainty. Each policy is an "act," each position a "state of nature," and the uncertainty criteria of decision theory apply.

- A decision maker who gives every position equal probability and maximizes expected utility applies the **Laplace criterion**. Since the expected utility is $(1/n)\Sigma u_i$, this rule chooses the same policy as the utilitarian criterion (Harsanyi, 1955).
- A decision maker who tries to make the worst case as good as possible applies the **maximin** rule and arrives at the egalitarian criterion (Rawls, 1971).

Example 12.14

In a society with three positions there are two policies: $P = (10, 6, 2)$ and $R = (5, 5, 5)$. The numbers are the utility of the person in each position.

1. **Laplace (Harsanyi).** The mean of P is $18/3 = 6$, the mean of R is 5. $6 > 5 \Rightarrow P$.
2. **Maximin (Rawls).** The worst position under P is 2, under R it is 5. $2 < 5 \Rightarrow R$.
3. **Nash social welfare.** The product for P is $10 \cdot 6 \cdot 2 = 120$, for R it is $5^3 = 125$. $125 > 120 \Rightarrow R$.

The same table recommends different policies depending on which uncertainty criterion is chosen. The utilitarian criterion looks at the size of the total, the egalitarian criterion at the worst case; in this example the Nash criterion chooses the policy close to equality.

The strength of the veil argument is that it reduces an argument about justice to the structure of a decision problem on which everyone can agree. The debate does not vanish; it only moves: the question "Which policy is just?" becomes "Which decision rule is reasonable under uncertainty?" Someone strongly risk averse is closer to maximin, a risk-neutral person to the Laplace criterion. These criteria are also basic tools in algorithmic fairness work that studies the fair division of resources and tasks.

12.4 Public goods and punishment

The common pasture is a problem of "taking": everyone takes too much from the common resource. A public good is a problem of "giving": everyone gives too little to the common project. The structure of the two problems is the same: an individual's decision creates an effect that does not enter their own calculation. This section solves the Public Goods Game from the start of the chapter theoretically, explains the results of laboratory experiments with a numerical model, and examines how punishment protects cooperation and where it falls short.

12.4.1 Public goods in the laboratory

A **public good** is a good whose use by one person does not reduce its use by another and from whose use no one can be excluded: street lighting, clean air, national defense, open-source software. Non-excludability means that those who do not contribute can benefit too. A person who benefits from others' contributions without contributing is a **free rider**, and the behavior is called **free riding**.

The Public Goods Game (12.1.1) is the laboratory model of this structure. Each token contributed returns 0.4 to the contributor and 1.6 to the group in total. The theoretical solution takes one line.

Theorem 12.1 (Contributing nothing is dominant)

In the four-player Public Goods Game with payoff $u_i = 10 - k_i + 0.4 \cdot \Sigma k$, $k_i = 0$ is a strictly dominant strategy for every player who cares only about their own money. In the unique Nash equilibrium everyone contributes 0 and receives 10; in the profile in which everyone contributes 10, everyone receives 16.

Proof. Whatever the other contributions are, raising player i's contribution k_i by 1 changes the payoff by $-1 + 0.4 = -0.6$: 1 token leaves the pocket and 0.4 returns from the pool. Since the payoff is a decreasing linear function of k_i, the maximum is attained at $k_i = 0$, and this does not depend on the others' contributions. If every player plays their dominant strategy the profile is (0, 0, 0, 0) and everyone receives 10. If everyone contributes 10 the pool is 40, becomes 64 when multiplied, and everyone receives 16. □

Theory gives a clear prediction: zero contribution. Laboratory experiments show a different picture (Ledyard, 1995):

- **First rounds:** about 40–60% of the endowment is contributed. Subjects are neither fully cooperative nor fully free riders.
- **Decline with repetition:** when the game is repeated for about ten rounds with the same group, contributions fall round by round and approach the free-riding level in the last rounds.
- **Who?** Fischbacher, Gächter, and Fehr (2001) asked subjects how much they would contribute for each possible value of the other members' average contribution, and thereby derived each subject's contribution schedule. About half the subjects are **conditional cooperators**: they want to contribute as much as the others do. About a third are free riders: they contribute 0 whatever the others do.

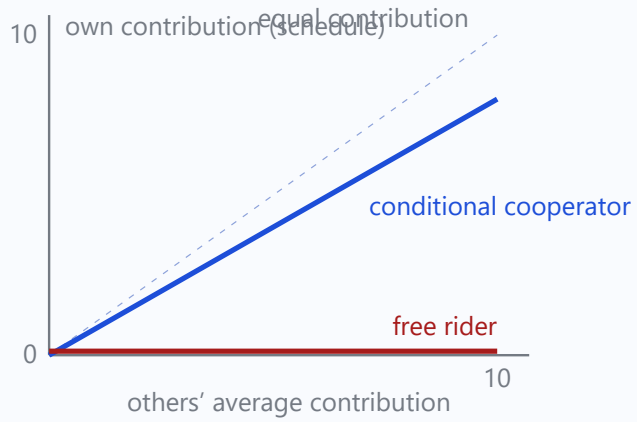


Figure 12.8: Contribution schedules. The free rider (red) contributes 0 in every case. The conditional cooperator (blue) raises the contribution as the others' average contribution rises, but stays slightly below the equal-contribution line (dashed).

Figure 12.8 shows the schedules of the two types. The mechanism of the decline can be read from this figure. Conditional cooperators contribute slightly below the average; the blue line lies below the dashed line. Free riders pull the average down. In the next round the conditional cooperators look at this lower average and again contribute slightly below it. The contribution erodes a little more each round.

12.4.2 Code trace: conditional()

The model is as follows: of four players, P1 always gives 0 (a free rider); P2, P3, and P4 are conditional cooperators and this round give the average of the others' contributions in the previous round. Initially (t = 0) the three cooperators have contributed 10 tokens each. The function returns the contribution list for every round.

```
def conditional(c, rounds):
  history = [c]
  for t in range(rounds):
    new = [0]                    # P1 always free-rides
    for i in range(1, len(c)):
      others = sum(c) - c[i]
      new.append(others / (len(c) - 1))
    c = new
    history.append(c)
  return history
```

The code runs line by line as follows:

- `history` stores the contribution lists of the rounds; the first element is the initial contributions.
- In each round the list `new` starts with `[0]` : P1 (index 0) is always a free rider.

- The inner loop runs over P2–P4 (indices 1, 2, 3). `others = sum(c) - c[i]` is the total contribution of the three players other than player `i` in the previous round. Dividing this total by `len(c) - 1 = 3` gives the average, which becomes the player's new contribution.
- At the end of the round `c` is replaced by the new contributions and appended to the history. The order matters: all players in a round look at the *previous* round's contributions; in the same round they do not see one another's new decisions.

Round	Sum of others for P2 (previous round)	Contributions	P1 payoff	P2 payoff
0	—	[0, 10, 10, 10]	22.0	12.0
1	0 + 10 + 10 = 20 ⇒ 20/3 ≈ 6.7	[0, 6.7, 6.7, 6.7]	18.0	11.3
2	13.3 ⇒ 4.4	[0, 4.4, 4.4, 4.4]	15.3	10.9
3	8.9 ⇒ 3.0	[0, 3.0, 3.0, 3.0]	13.6	10.6
4	5.9 ⇒ 2.0	[0, 2.0, 2.0, 2.0]	12.4	10.4
5	4.0 ⇒ 1.3	[0, 1.3, 1.3, 1.3]	11.6	10.3

Table 12.11: Trace of `conditional([0, 10, 10, 10], 5)`. Payoff $u = 10 - k + 0.4 \cdot \Sigma k$.

Table 12.11 shows each round in detail. In going from t = 0 to round 1, P2 looks at the sum of the others: P1's 0, P3's 10, and P4's 10, a total of 20. The average is *20/3 ≈ 6.7*. P3 and P4 make the same calculation. So in round 1 the contributions are [0, 6.7, 6.7, 6.7]. In round 2 the sum of the others for P2 is *0 + 6.7 + 6.7 ≈ 13.3*, the average 4.4; and so it goes on.

The pattern can be written in general. If the cooperators contribute x in one round, then in the next round the sum of the others for each is $0 + x + x = 2x$ and the average is $2x/3$. Each round the contribution falls to two thirds of the previous value (shrinks by a third):

> $$x_t = 10 \cdot (2/3)^t: \quad 10 \to 6.7 \to 4.4 \to 3.0 \to 2.0 \to 1.3$$

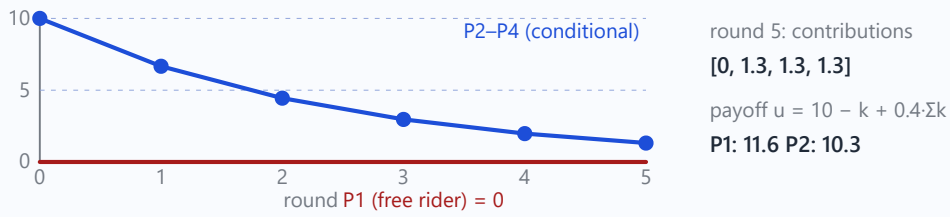


Figure 12.9: The conditional cooperators' contribution (blue) falls from 10 to 1.3 in five rounds; the free rider P1 (red) always gives 0. In round 5 the payoffs are 11.6 for P1 and 10.3 for P2.

Figure 12.9 shows the decline. In five rounds the contribution falls from 10 to 1.3: a single free rider dissolves the majority's cooperation. The payoff columns are instructive too. In every round the free rider P1 earns more than the cooperators: at t = 0, 22 against 12. Free riding pays every round; the cooperators' "punishment" is to cut their contributions. But this punishment hits everyone,

themselves included: in round 5 the cooperators get 10.3 instead of 12. This simple model captures the qualitative features of the laboratory decline: a high start, round-by-round erosion, and an approach to the free-riding level in the last rounds.

Common mistake

"Since contributions fell, the subjects learned to free ride." In this model nobody learns anything. The cooperators never change their rule: each round they contribute the others' average. The decline arises by itself from the combination of the conditional-cooperation rule and a single free rider. In experiments, too, an important part of the decline is thought to come from this mechanism.

12.4.3 With and without punishment: Fehr and Gächter

If the only weapon in the conditional cooperators' hands is cutting their contributions, they punish themselves as they punish the free rider. Fehr and Gächter (2000) gave subjects a second weapon: direct punishment.

The design of the experiment is as follows. Groups of 4 play the public goods game; each token contributed returns 0.4 tokens to the contributor. At the end of each round the contributions are announced (anonymously). Then each subject may assign punishment points to the other members, paying from their own money; the payoff of whoever receives the points falls. Each group plays 10 rounds with and without punishment. The experiment was run in two forms: fixed groups, in which the same people play together every round, and groups reshuffled every round.

- **Without punishment:** contributions are high in the first rounds and then fall toward zero. The dynamics of 12.4.2 are seen here as well.
- **With punishment:** contributions rise; in fixed groups they approach full cooperation. Those who contribute furthest below the group average receive the most punishment.

The most surprising finding concerns the groups reshuffled every round. In these groups the probability of meeting the same person again is almost nil. Punishing gives the punisher nothing: the punished person's future behavior does not affect the punisher's future groups. Even so, subjects punished. The behavior of punishing, at one's own cost, someone who violates a norm, when there is no material gain to oneself, is called **altruistic punishment** (Fehr and Gächter, 2002).

The theory's prediction is clear. For someone who cares only about money, punishing is a pure cost; so nobody punishes. Knowing this, everyone behaves as if there were no possibility of punishment and again contributes 0. In the experiment punishment was both used and contributions stayed high: both of the theory's predictions failed. Inequity aversion and reciprocity explain this behavior: the free rider is ahead and acts with bad intentions; punishing the free rider reduces the inequality and answers the intention.

12.4.4 Code trace: punish()

Can punishment make free riding unprofitable? The function below computes the players' payoffs given contributions and punishment points. The rule is this: the punisher pays 1 token for each point, and the punished loses 3 tokens for each point. A punishment point harms the punished three times as much as it costs the punisher; punishment is effective but also expensive for the punisher.

```
def punish(c, p):        # p[i][j]: i's points to j
    n = len(c)
    share = 1.6 * sum(c) / n
    u = [10 - c[i] + share for i in range(n)]
    for i in range(n):
        for j in range(n):
            u[i] -= p[i][j]       # costs 1 per point
            u[j] -= 3 * p[i][j]   # target loses 3
    return u
```

The code runs line by line as follows:

- `p[i][j]` is the number of punishment points player `i` gives to player `j`; `p` is a points matrix.
- `share` is the amount from the pool that falls to each player: the total contribution is multiplied by 1.6 and divided by `n`.
- `u` holds the payoffs before punishment: the tokens kept, `10 - c[i]`, plus the pool share.
- The nested loop runs over every (`i`, `j`) pair. The number of points is deducted from the punisher `i`'s payoff, and 3 times the points from the punished `j`'s payoff.

Let the call be `punish([0, 10, 10, 10], p)`: P1 is a free rider and P2–P4 contribute fully. Each cooperator gives P1 2 punishment points; in the points matrix `p[1][0] = p[2][0] = p[3][0] = 2` and everything else is 0.

Step	Operation	u = [P1, P2, P3, P4]
Start	*share = 1.6 · 30 / 4 = 12*; P1: *10 − 0 + 12*, the others *10 − 10 + 12*	[22, 12, 12, 12]
i = 0	P1 punishes nobody	[22, 12, 12, 12]
i = 1	P2 pays 2 for 2 points; P1 loses 2 · 3 = 6	[16, 10, 12, 12]
i = 2	P3 also pays 2; P1 loses 6 more	[10, 10, 10, 12]
i = 3	P4 also pays 2; P1 loses 6 more	[4, 10, 10, 10]

Table 12.12: Trace of `punish([0, 10, 10, 10], p)`; each cooperator gives P1 2 points.

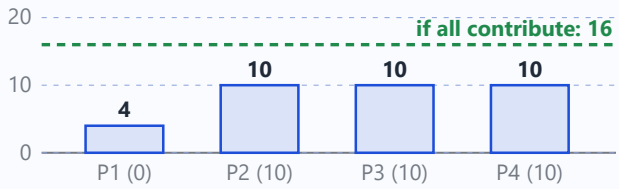


Figure 12.10: Payoffs after punishment. The free rider P1 gets 4; the punishing cooperators get 10 each. Had everyone contributed, everyone would have received 16 (green dashed line).

Table 12.12 and Figure 12.10 give the result. Before punishment the free rider earned 10 more than the contributors (22 against 12). After P3's punishment (i = 2) the free rider's advantage ends: P1 and P2, P3 are equal at 10. After three punishments the free rider gets 4. Had P1 contributed and nobody punished, everyone would have contributed 10 and P1 would have received 16. If the threat of punishment is credible, contributing pays: 16 > 4. But the punishers have also each lost 2; punishment is costly to everyone. This raises a new question: who will do the punishing?

12.4.5 Second-order free riding and antisocial punishment

Punishment deters free riding, but punishing is also costly. Let the same `punish()` function be run under three different punishment arrangements (Table 12.13).

Punishers	P1	P2	P3	P4
Nobody	22	12	12	12
P2, P3, P4	4	10	10	10
P2, P3 (P4 does not)	10	10	10	12

Table 12.13: Contributions [0, 10, 10, 10]; a punishment point costs 1 and removes 3. Each punisher gives P1 2 points.

The last row is important. P4 contributed but did not punish. The punishment by P2 and P3 has already erased the free rider's advantage; P4 enjoys this benefit without bearing the cost of punishing and receives the highest payoff in the group (12). Punishment is itself a public good: its cost falls on the punisher, its benefit (deterring the free rider, higher contributions in the future) on everyone. A cooperator who does not punish is a **second-order free rider**, and the behavior is **second-order free riding**. Punishing those who do not punish looks like a solution, but that punishment is costly too and the problem moves up one level: third-order free riding.

For players who care only about money the result is found by backward induction. In the last round punishing is only a cost, because there is no future round for the punishment to deter; nobody punishes. Players who know this do not contribute in the last round. In the round before, the behavior of the future rounds is already settled; punishing is again only a cost. This logic runs all the way back: the threat is not credible and contributions fall to 0. In experiments, however, people do punish (12.4.3): altruistic punishment partly overcomes the problem of second-order free riding.

It has also been found that punishment does not work the same way everywhere. Herrmann, Thöni, and Gächter (2008) ran the same public goods experiment with punishment in 16 cities, Istanbul among them. In some cities subjects punished not only low contributors but also *high* contributors. Punishing those who comply with norms is called **antisocial punishment**. Where antisocial punishment was common, the cooperation-raising effect of punishment was largely erased. The authors found that antisocial punishment was more frequent in societies where civic norms and the rule of law are weak. One possible cause of antisocial punishment is revenge: a free rider punished in the previous round retaliates against the high contributors they believe punished them.

The lesson of this finding is that punishment is an institution, and how it works depends on the norms around it. When designing a platform or an institution, the rule "users may punish one another" does not by itself produce cooperation; who uses the rule, against whom, and on what grounds also matters. Ostrom's principles of graduated sanctions and accountable monitoring (12.3.4) can also be read as institutional safeguards that limit the abuse of punishment.

Common mistake

"Punishment always increases cooperation." Punishment is costly and gives rise to second-order free riding; antisocial punishment can erase its effect. Whether the option to punish raises cooperation depends largely on whether those who use it target the norm violators rather than the norm followers.

12.4.6 Back to the Public Goods Game

The two questions at the start of the chapter can now be answered. The first was the contribution predicted by theory and by experiments. Theory's answer was given in Theorem 12.1: 0 tokens. Each token contributed returns only $1.6/4 = 0.4$ to the contributor; whatever the others do, not contributing is better by 0.6 per token. In reality, in repeated experiments contributions in the first rounds are about 40–60% of the endowment and decline as rounds go on (Ledyard, 1995). When participants can punish one another at their own expense, contributions stay high (Fehr and Gächter, 2000).

The second question was the game's "Punished contribution" scenario: anyone contributing fewer than 3 tokens is fined 2 tokens. Here the punishment comes not from the players but from the rule itself; the punisher bears no cost. This rule changes the players' incentives.

Example 12.15

The payoff is $u_i = 10 - k_i + 0.4 \cdot \Sigma k - 2 \cdot [k_i < 3]$. The square bracket is 1 if the condition inside is true and 0 otherwise.

1. **Contributing more than 3.** Each token above 3 costs 0.6, as in the game without the fine; the fine status does not change. Contributing more than 3 is worse than contributing 3.

2. **Contributing 1 or 2.** One loses 0.6 per token and also pays the 2-token fine. This is worse than contributing 0.
3. **Comparing 0 and 3.** Contributing 3 costs $3 \cdot 0.6 = 1.8$ relative to contributing 0 but saves the 2-token fine. The net difference is $2 - 1.8 = +0.2$: contributing 3 is better.
4. **Result.** Whatever the others contribute, 3 is best: 3 is a dominant strategy. In equilibrium everyone contributes 3; the pool is 12, becomes 19.2 when multiplied, the share per player is 4.8; the payoff is $10 - 3 + 4.8 = 11.8$.

Scenario	Equilibrium contribution	Payoff per player
Baseline	0	10
Punished (2 fine for contributing under 3)	3	11.8
Optimum (everyone 10)	10	16

Table 12.14: Scenarios of the Public Goods Game.

Table 12.14 compares the three situations. The fine rule, like the Pigouvian tax on the common pasture, is an institution that changes incentives: it moves the equilibrium from 0 to 3 and raises everyone's payoff from 10 to 11.8. But it does not lead to the optimum (contribution 10, payoff 16). The effect of the rule depends on the size of the threshold and the fine. Had the fine been 1 token, contributing 3 would cost 1.8 yet save only 1 token; not contributing would be 0.8 better and the equilibrium would again be 0. The general rule is this: for a threshold e and a fine c, contributing e becomes dominant when $c > 0.6 \cdot e$. To force full contribution (10) the fine must exceed 6.

12.5 Threshold projects

In the public goods game every token contributed enlarges the pool a little; the relationship between contribution and benefit is linear. In many real projects, however, the benefit appears only when a certain threshold is reached. A bridge built halfway is useless; a crowdfunding campaign that misses its target produces no product; a petition that does not collect enough signatures is not taken up. This structure turns the contribution problem into an entirely different game.

12.5.1 Contribution becomes a coordination game

Definition 12.12

Threshold public good (or **threshold project**): a public good from which everyone receives a benefit if the total contribution reaches a certain threshold, and the contributions are wasted if it does not. In the threshold-project scenario of the Public Goods Game, 4 players start with 10 tokens each; if the total contribution reaches 22 everyone receives 12 extra tokens, and if it does not the contributions are lost:

$$u_i = 10 - k_i + 12 \cdot [\Sigma k \geq 22]$$

The equilibria of this game are very different from those of the linear public goods game. Contributing nothing is no longer dominant: when the threshold is missed by a hair, the player who supplies the missing tokens gains 12. But contributing is not dominant either: if the threshold cannot be reached, every contribution is wasted. A player's best response depends on what the others contribute. This turns the game into a **coordination game**: the right decision for the players depends on their sharing the same expectation as the others.

There are two kinds of equilibrium.

Equilibrium 1: nobody contributes. If everyone contributes 0, the most a single player can contribute is 10 tokens and the threshold of 22 cannot be reached. A contributor only loses. Nobody wants to deviate; everyone receives 10, 40 in total.

Equilibrium 2: the total is exactly 22. If the total contribution is exactly 22, every contributor is *pivotal*: withdrawing even 1 token drops the total to 21, the threshold is missed, and 12 tokens are lost. Contributing extra is pointless too: the threshold is already met and the excess is wasted. For example, in the profile (6, 6, 5, 5) the payoffs are (16, 16, 17, 17), 66 in total.

Which equilibrium is played? Theory does not say. The zero equilibrium is a safe equilibrium that exposes nobody to risk; the threshold equilibrium is better for everyone but requires trust. A contributor whose partners do not contribute burns the contribution: a player who contributes 6 falls to 4 instead of 10 if the others do not contribute. Without trust the zero equilibrium may be chosen. This is the same dilemma as in the Stag Hunt: the cooperative equilibrium is Pareto dominant, but a lone cooperator suffers a big loss.

A second coordination problem is the multiplicity of equilibria that reach the threshold. The contribution profile (10, 10, 2, 0) is also an equilibrium; the payoffs are (12, 12, 20, 22). Every profile whose total is exactly 22 is an equilibrium, and in all of them the group's total payoff is the same: $40 - 22 + 4 \cdot 12 = 66$. The threshold equilibria are efficient but may be unequal. Who contributes how much is a bargaining problem: every player wants the others to contribute more.

In real-life threshold projects a mechanism that eases these problems is common: the **refund**. In crowdfunding campaigns that miss their target the money is returned to the contributors. Then if the threshold is not met the contribution is not burned; the risk of "contributing and burning it" disappears. The player's contribution is spent only when the threshold is met, that is, the contribution is paid only when the 12 is received in return. This rule erases the safety advantage of the zero equilibrium and makes coordination easier.

12.5.2 Which contribution profiles are equilibria?

To check whether a profile is an equilibrium, we ask whether any player can raise their payoff by deviating alone. In a threshold project there are two kinds of deviation: lowering the contribution (if the threshold is met) and raising it (if it is not). Let us check five profiles in turn.

Example 12.16

1. **(0, 0, 0, 0).** If P1 alone contributed 10, the total would be 10 < 22 and the 10 tokens would be burned. No player can reach the threshold alone; nobody wants to deviate. It is a Nash equilibrium; everyone's payoff is 10.
2. **(6, 6, 5, 4).** The total is 21, the threshold is missed; the payoffs are (4, 4, 5, 6). If P4 contributes 5 the total becomes 22 and P4 gets $10 - 5 + 12 = 17 > 6$. It is not an equilibrium.
3. **(6, 6, 5, 5).** The total is 22, the payoffs are (16, 16, 17, 17). If P1 drops to 5 the total becomes 21 and P1 gets $10 - 5 = 5 < 16$. The same holds for the other players; contributing more also pays in vain. It is a Nash equilibrium; the total is 66.
4. **(7, 6, 5, 5).** The total is 23. If P1 drops to 6 the total becomes 22, the threshold is still met, and P1 gains 1 token (16 instead of 15). This profile is not an equilibrium.
5. **(10, 10, 2, 0).** The total is 22. If P3 withdraws (contributes 0) the total becomes 20 and the threshold is missed: P3 gets 10 instead of 20. P1 and P2 are pivotal too. If P4 contributes, P4 only loses. It is a Nash equilibrium; the payoffs are (12, 12, 20, 22).

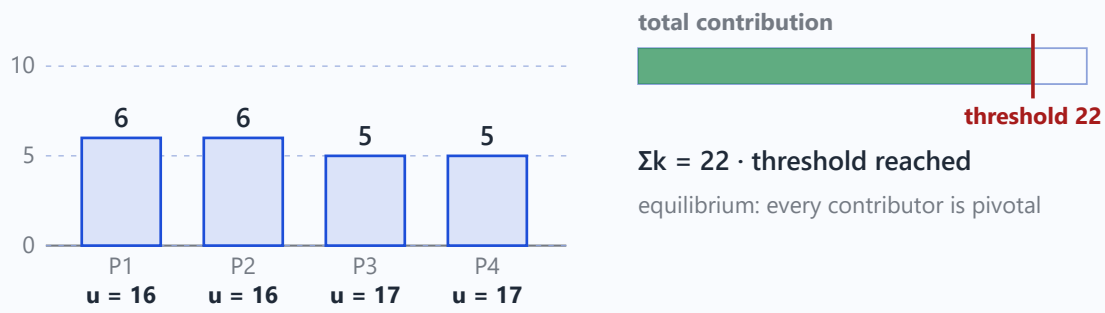


Figure 12.11: The contribution profile (6, 6, 5, 5). The total contribution is exactly 22 and the threshold is reached; the payoffs are (16, 16, 17, 17). Every contributor is pivotal: if anyone withdrew 1 token the

threshold would be missed.

Figure 12.11 shows the (6, 6, 5, 5) equilibrium. A general conclusion follows from these five checks: in this game the Nash equilibria are of exactly two kinds. Either everyone contributes 0, or the total contribution is exactly 22. If the total is between 0 and 22 the threshold is not met; any player who contributes gains by cutting the contribution to 0, because it is wasted anyway. If the total exceeds 22, any contributing player reduces the contribution by 1, the threshold is still met, and the player gains 1 token. Only the profiles with total 0 and total exactly 22 remain; 12.5.1 showed that both kinds are equilibria. Scanning all possible profiles by computer confirms this: of the $11^4 = 14{,}641$ profiles in which each player can contribute between 0 and 10, all 850 profiles with total exactly 22, and only those, together with the zero profile, are equilibria. The efficient equilibria are many; who contributes what is a bargaining and coordination problem.

Common mistake

"A threshold project has a single equilibrium." Both zero contribution and every profile whose total is exactly 22 are equilibria. Moreover, "equal contributions (5 each)" is not a special equilibrium; an equal split is not even possible with whole tokens, and very unequal profiles such as (10, 10, 2, 0) are also equilibria.

12.6 Collective information

In the earlier sections the interests of the group members conflicted: when one took more, the others lost. In this section everyone's interest is the same: finding the right answer to a question. The difficulty is that each member knows the answer only in part. Whether the group can pool this scattered knowledge depends on how information is shared. Three models treat the question from different angles: the wisdom of crowds (averaging estimates), information cascades (decisions made in sequence, each following the last), and the Condorcet jury theorem (majority voting).

12.6.1 The wisdom of crowds

In 1906 Francis Galton came across an interesting competition at a livestock fair in England. Visitors bought tickets to guess the weight of an ox that was to be slaughtered and dressed. After the contest Galton collected and examined 787 valid guesses. The median of the guesses (the one in the middle when sorted) was 1,207 pounds; the ox's actual weight was 1,198 pounds: an error of less than 1% (about 0.75%). Among the participants were butchers and farmers, but most were non-experts; the combination of their guesses was strikingly accurate (Galton, 1907).

This phenomenon is known as the **wisdom of crowds**. The basic explanation is that independent errors cancel out. One person overestimates the weight and another underestimates it; if the errors point in different directions and are independent of one another, they largely cancel in the average.

Scott Page turned this intuition into an exact identity.

Theorem 12.2 (Diversity prediction theorem)

Let the true value be θ, the estimates of n people $x_1, \ldots, x_n$, and their average $\bar{x}$. Then

$$(\bar{x} - \theta)^2 = (1/n)\,\Sigma\,(x_i - \theta)^2 - (1/n)\,\Sigma\,(x_i - \bar{x})^2$$

In words: the crowd's error = the average of the individual squared errors − the diversity of the predictions (Page, 2007). Diversity is the average of the squared deviations of the estimates from their own mean.

Proof. Split each individual error into two parts: $x_i - \theta = (x_i - \bar{x}) + (\bar{x} - \theta)$. Square it: $(x_i - \theta)^2 = (x_i - \bar{x})^2 + 2(x_i - \bar{x})(\bar{x} - \theta) + (\bar{x} - \theta)^2$. Sum over all i and divide by n. In the middle term $(\bar{x} - \theta)$ is the same for every i and comes outside; what remains is $\Sigma(x_i - \bar{x})$. This sum is zero, because the deviations from the mean always add up to zero: $\Sigma x_i - n \cdot \bar{x} = 0$. The last term, summed n times and divided by n, is $(\bar{x} - \theta)^2$. So we obtain: average individual error = diversity + the crowd's error; rearranging gives the theorem. □

Example 12.17

The true value is 50; five people's estimates are 40, 45, 55, 60, 70.

1. **Average.** $(40 + 45 + 55 + 60 + 70)/5 = 270/5 = 54$.
2. **The crowd's error.** $(54 - 50)^2 = 16$.
3. **Individual squared errors.** $(x - 50)^2$: 100, 25, 25, 100, 400. Total 650, average 130.
4. **Diversity.** $(x - 54)^2$: 196, 81, 1, 36, 256. Total 570, average 114.
5. **Check.** $130 - 114 = 16$ ✓.

The crowd's error (16) is far smaller than the error of an average member (130); the difference is exactly the diversity (114).

The consequences of the theorem are strong. Since diversity is never negative, the crowd's error is never larger than the error of an average member: the crowd always predicts at least as well as its average member. The greater the diversity, the greater the gain. A smaller example shows the same relation: let the true value be 10 and the estimates 8, 12, 16. The average is 12 and the crowd's error is 4. The individual squared errors are 4, 4, 36 (average 44/3) and the diversity terms are 16, 0, 16 (average 32/3); the difference is $12/3 = 4$ ✓.

The theorem is an identity and always true; but the crowd's being truly wise depends on two conditions: **independence** and **diversity**. If the estimates are not independent of one another, the errors do not cancel but pile up in the same direction. If everyone follows everyone else, the estimates

converge, diversity collapses, and the crowd can be as wrong as a single member. This collapse arises even among rational individuals in information cascades (12.6.2).

Common mistake

"The crowd is better than its best member." The theorem does not say this. The crowd is at least as good as the *average* member; it can be worse than the best member. In the example the error of those who guessed 45 and 55 is 25; the crowd's 16 beats them, but had there been someone who knew the true value exactly, that person's error would have been 0.

12.6.2 Information cascades

In a new city two restaurants stand side by side. One has a queue; the other is empty. Most people join the queue: "that many people can't be wrong." But each of the people in the queue may have thought the same thing. The first few may have entered that restaurant by chance and the later ones followed them. In that case the queue carries almost no information about the restaurant's quality. Bikhchandani, Hirshleifer, and Welch (1992) studied this phenomenon with an exact model.

The model. There are two urns. Urn A contains 2 a-balls and 1 b-ball, urn B contains 1 a-ball and 2 b-balls (Figure 12.12). One of the urns is chosen by a coin toss; nobody knows which. In turn, each person secretly draws a ball from the urn, sees it, and puts it back. Then they hear the earlier people's guesses (not their balls) and say aloud whether they think the urn is A or B. A correct guess is rewarded.

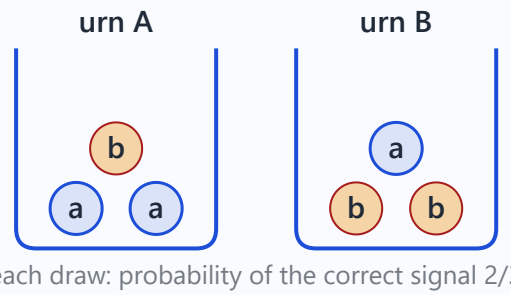


Figure 12.12: The two urns. In urn A 2/3 of the balls are a, in urn B 2/3 are b; each draw shows the urn's own color with probability 2/3.

Each ball is the urn's own color with probability 2/3; this is why the ball is called a **signal**: it carries noisy but useful information about the urn. Everyone's question is the same: given the information at hand, what is the probability that the urn is A?

Bayesian calculation. At the start the two urns are equally likely: $P(A) = 1/2$. This is the prior probability. After an a signal is seen, the posterior probability is computed with Bayes' rule:

$$P(A \mid a) = P(a \mid A) \cdot P(A) / [P(a \mid A) \cdot P(A) + P(a \mid B) \cdot P(B)] = (\tfrac{2}{3} \cdot \tfrac{1}{2}) / (\tfrac{2}{3} \cdot \tfrac{1}{2} + \tfrac{1}{3} \cdot \tfrac{1}{2}) = 2/3$$

The way to simplify the calculation is to think in terms of odds. The odds in favor of A start at 1 to 1. An a signal has probability 2/3 under A and 1/3 under B; so each a doubles the odds in favor of A. Each b halves them. With independent signals the order does not matter; only how many more a's than b's there are matters. When there are f more a's than b's the odds are 2^f to 1 and

$$P(A) = 2^f / (2^f + 1)$$

Example 12.18

1. **f = 0** (no signals, or equal numbers of a and b): $P(A) = 1/(1 + 1) = 1/2$.
2. **f = 1** (one more a): $P(A) = 2/3$.
3. **f = 2** (two more a's): $P(A) = 4/5 = 0.8$.
4. **f = −2** (two more b's): $P(A) = 1/(4 + 1) = 0.2$.
5. **Signals a, a, b** (f = 1): the long way, $P(A) = (\tfrac{2}{3} \cdot \tfrac{2}{3} \cdot \tfrac{1}{3}) / (\tfrac{2}{3} \cdot \tfrac{2}{3} \cdot \tfrac{1}{3} + \tfrac{1}{3} \cdot \tfrac{1}{3} \cdot \tfrac{2}{3}) = (4/27)/(4/27 + 2/27) = 2/3$. The shortcut gives the same result.

What do people do? The first person has only their own signal; on seeing a they say A, on seeing b they say B. Their guess reveals their signal exactly. The second person infers the first person's signal from the first guess and combines it with their own. If the two signals agree, they say so. If they differ, $f = 0$ and the two urns are equally likely; by the model's convention, in a tie the person follows their own signal. So the second person's guess also reveals their signal.

The cascade. If the first two people saw a and said A, the third person infers two a's from the earlier guesses: $f = 2$. If their own signal is b, $f = 2 - 1 = 1$ and $P(A) = 2/3 > 1/2$. It is rational to ignore their own signal and say A. Had their own signal been a, they would also say A. So the third person's guess is A whatever their signal, and carries no information about it. The fourth person can learn nothing from the third saying A; they are in exactly the same position as the third and say A for the same reason. Everyone after that says A.

Definition 12.13

Information cascade: a person finds the information inferred from the behavior of those before them stronger than their own private information, ignores their own information and follows the others, so that their behavior carries no new information for those after. In the two-urn model the cascade begins when the difference inferred from earlier guesses reaches $|f| = 2$.

Cascades have two important properties. First, they **can be wrong**. Even if the true urn is B, the probability that the first two people both draw a is $⅓ \cdot ⅓ = 1/9$; in that case the whole crowd falls into a wrong A cascade, and it is not corrected no matter how many people later see b. Second, they are **fragile**. A cascade rests on only two signals; the hundreds of people saying A really carry the information of two balls. A single piece of public information (for example, two b balls that everyone can see) can break it. The sudden reversals of fashion trends, product reviews, and financial panics are explained by this fragility: a long-running trend can be completely reversed by a small new piece of information.

The model has also been tested in the laboratory. Anderson and Holt (1997) ran this urn experiment with students. In most of the situations in which cascades could form (that is, in which the difference inferred from the guesses reached 2), cascades did form: subjects followed those before them even when it went against their own signals.

12.6.3 Code trace: cascade()

The cascade model can be written as a function of a few lines. The function takes the sequence of signals and returns each person's guess. The variable `f` is the a − b difference inferred from the earlier guesses.

```
def cascade(signals):
    f = 0                  # a - b inferred from guesses
    guesses = []
    for s in signals:
        if f >= 2: t = 'A'     # A cascade
        elif f <= -2: t = 'B'  # B cascade
        else:
            t = 'A' if s == 'a' else 'B'
            f += 1 if s == 'a' else -1
        guesses.append(t)
    return guesses
```

The code runs line by line as follows:

- `f` starts at 0: there are no guesses yet.
- If `f >= 2` there is an A cascade: the person says A whatever their own signal. If `f <= -2` there is a B cascade: the person says B. In these two cases `f` does not change, because a guess inside a cascade carries no information for those after.
- Otherwise ($|f| \le 1$) the person follows their own signal. When f = 1, their own a makes f = 2 and their own b makes f = 0, and in a tie they follow their own signal; f = −1 is symmetric. In this case the guess reveals the signal; later people can infer it, so `f` is updated.

Person	Signal	f from earlier guesses	P(A) with own signal	Guess	Next f
1	a	0	2/3	A	1

Person	Signal	f from earlier guesses	P(A) with own signal	Guess	Next f
2	a	1	0.8	A	2
3	b	2	2/3	A (cascade)	2
4	b	2	2/3	A	2
5	b	2	2/3	A	2
6	b	2	2/3	A	2

Table 12.15: Trace of `cascade(['a', 'a', 'b', 'b', 'b', 'b'])`; the true urn is B.



Figure 12.13: A cascade of six people. The first two see a and say A; once f = 2 the cascade begins, and the next four people ignore their own signals and say A even though they see b. The true urn is B.

Table 12.15 and Figure 12.13 show the trace. Person 1 sees a; since f = 0 they follow their own signal and say A. Their guess reveals their signal: f = 1, *P(A) = 2/3* for the next person. Person 2 also sees a; with their own a, *P(A) = 0.8* and they say A; f = 2. Person 3 sees b; but the two a's inferred from the earlier guesses combined with their own b give *P(A) = 2/3*, and they say A. The cascade has begun; f no longer changes. Persons 4, 5, and 6 also see b and say A by the same calculation; their guesses carry no information either.

The function returns `['A', 'A', 'A', 'A', 'A', 'A']`: everyone said A, yet 4 of the 6 signals are b. Had the signals been pooled, $f = 2 - 4 = -2$ and *P(A) = 0.2*; the correct answer, B, would have been found. Each person acted correctly by Bayes with the information they had; the problem is that information does not accumulate. There were four b signals, but none was reflected in the guesses, so they were lost.

Common mistake

"Those who follow the cascade are irrational," or "they conform out of social pressure." In the model nobody feels social pressure and everyone calculates correctly by Bayes. Person 3's saying A is the best guess given the information at hand. The problem lies not in the individuals but in how information is shared: people announce their guesses, not their signals.

12.6.4 The Condorcet jury theorem

In cascades people decide in turn and follow one another. In a jury, by contrast, everyone votes at the same time and independently; the decision is made by majority. There is a single right answer (is the defendant guilty or not?). The question is how the probability that the majority vote finds this right answer changes as the jury grows. This question differs from the Condorcet paradox, in which preferences go in a cycle: the paradox asks "is the group's preference consistent?"; the jury theorem asks "does the group find the truth?"

Theorem 12.3 (Condorcet jury theorem)

If n (an odd number of) voters vote independently and each picks the right answer with probability $p > 1/2$, the probability that the majority is right increases with n and tends to 1 as $n \to \infty$ (Condorcet, 1785). If $p < 1/2$ the opposite holds: as the crowd grows, the majority is almost surely wrong.

To compute the probability that the majority is right we use the **binomial probability**. The probability that exactly k of the n voters vote correctly is the product of the number of ways to choose the k correct voters and the probability of a single such choice:

$$P(\text{exactly } k \text{ correct}) = C(n, k) \cdot p^k \cdot (1 - p)^{n - k}$$

Here $C(n, k)$ is the number of ways to choose k people out of n (for example $C(5, 3) = 10$). p^k is the probability that all k chosen people are right; $(1 - p)^{n - k}$ is the probability that all the rest are wrong; thanks to independence the probabilities multiply. If the number of correct votes is greater than $n/2$, the majority is right. Therefore

$$P_n = \Sigma_{k > n/2}\, C(n, k) \cdot p^k \cdot (1 - p)^{n - k}$$

Example 12.19

A jury of three, $p = 0.6$. A majority needs at least 2 correct votes.

1. **All three correct.** $C(3, 3) \cdot 0.6^3 = 0.216$.
2. **Exactly two correct.** The wrong voter can be chosen in 3 different ways: $C(3, 2) \cdot 0.6^2 \cdot 0.4 = 3 \cdot 0.36 \cdot 0.4 = 0.432$.
3. **Total.** $P_3 = 0.216 + 0.432 = 0.648$.

A single voter is right with probability 60%; the majority of a three-person jury is right with probability 64.8%.

n	p = 0.6	p = 0.4
1	0.600	0.400
3	0.648	0.352
5	0.683	0.317
11	0.753	0.247
51	0.926	0.074
101	0.979	0.021

Table 12.16: Probability P_n that the majority decides correctly.

Table 12.16 shows the two faces of the theorem. With *p = 0.6* each voter is only slightly informed, yet a majority of 101 is right with probability 97.9%. With *p = 0.4* the table reverses symmetrically: with 101 voters the majority is right with probability only 2.1%. The reason is the law of large numbers: as the number of voters grows, the fraction of correct votes approaches *p*. If *p > 1/2* this fraction exceeds a half and the majority is right; if *p < 1/2* it stays below a half and the majority is wrong. The two columns of the table add up to 1 in every row, because the number of correct votes at *p = 0.4* has the same distribution as the number of wrong votes at *p = 0.6*.

12.6.5 Code trace: jury()

The binomial sum is computed with a short function. Python's `math.comb(n, k)` gives *C(n, k)*.

```
from math import comb
def jury(n, p):
    total = 0
    for k in range(n // 2 + 1, n + 1):
        term = comb(n, k) * p**k * (1 - p)**(n - k)
        total += term
    return total
print(round(jury(5, 0.6), 4))
```

The code runs line by line as follows:

- `range(n // 2 + 1, n + 1)` produces the numbers of correct votes that give a majority. `n // 2` is integer division; for `n = 5` it is 2, so the loop takes `k = 3, 4, 5`.
- For each `k`, `term` is the probability that exactly `k` people are right, and it is added to `total`.
- The last line rounds `jury(5, 0.6)` to four decimals and prints it.

k	C(5, k)	0.6^k	0.4^{5-k}	term	total
3	10	0.2160	0.1600	0.3456	0.3456
4	5	0.1296	0.4000	0.2592	0.6048

k	C(5, k)	0.6^{k}	0.4^{5-k}	term	total
5	1	0.0778	1	0.0778	0.6826

Table 12.17: Trace of `jury(5, 0.6)`.

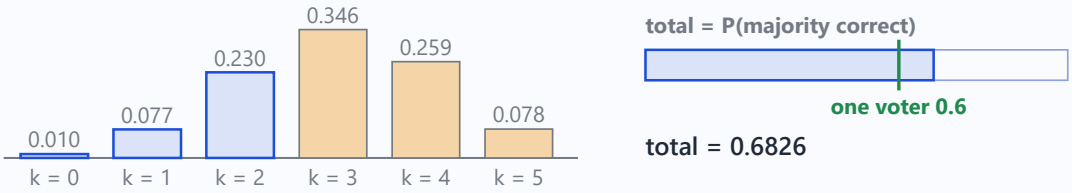


Figure 12.14: Distribution of the number of correct votes in a jury of five ($p = 0.6$). The orange bars (k = 3, 4, 5) are the cases in which the majority is right; their sum is 0.6826. A single voter's probability is 0.6.

Table 12.17 gives the three terms one by one; Figure 12.14 shows the whole distribution. The most likely outcome is exactly 3 correct votes (0.3456). The probabilities of k = 0, 1, 2 (a wrong majority) are 0.010, 0.077, and 0.230, summing to 0.3174; the two sums add up to 1. The program prints `0.6826`: $P_5 = 0.6826 > 0.6$. Five independent voters of moderate competence decide better than a single voter.

12.6.6 When does the jury theorem break down?

The theorem has two assumptions: the votes are independent, and each voter's probability of finding the truth is greater than a half. A third, implicit assumption is that voters vote honestly on what they know. When any of these fails, the theorem's conclusion fails too.

Dependence. In a jury of 5, if 4 people follow the vote of an opinion leader, the majority is always that person's vote and the probability of being right is 0.6. With independent votes it was 0.683. Dependence reduces the jury to a single voter; the crowd's advantage is lost. Information cascades and common information sources (everyone reading the same news story) are the main mechanisms that break independence.

Competence. On a question where $p < 1/2$, for example when a clue misleads everyone in the same direction, error becomes certain as the crowd grows: with $p = 0.4$ a majority of 101 is right with probability only 2.1%. Enlarging the crowd strengthens not the information but the shared mistake.

Strategic voting. If voters take into account that their vote matters only when it can change the outcome, that is, when it is pivotal, voting against their own signal can be rational (Austen-Smith and Banks, 1996). For example, under a unanimity rule a juror's vote is pivotal only when all the other members say "guilty." It follows that all the other members saw a sign of guilt; this information can be strong enough to override the juror's own "innocent" sign. By this logic jurors may drift to a "guilty" vote, and the unanimity rule can raise the probability of convicting an innocent person (Feddersen and Pesendorfer, 1998). A rule designed to protect the innocent can work in reverse because of strategic behavior.

The wisdom of crowds, cascades, and the jury theorem teach the same lesson: a group's knowledge is valuable to the extent that its members' independent information can come together. Averaging, sequential decisions, and majority voting are three different mechanisms for combining information; all three fail when independence is broken.

Common mistake

"The jury theorem and the Condorcet paradox are the same." Both bear Condorcet's name but answer different questions. The theorem assumes a right answer exists and concerns the probability that the majority finds it. The paradox looks for no right answer; it shows that, even when individual preferences are consistent, majority preferences can cycle (A > B, B > C, C > A).

12.7 Common mistakes in collective decisions and example problems

Because the results of the models in the second half of the chapter resemble one another, they are easily, and sometimes wrongly, generalized. The list below gives the most frequent misconceptions together with the correct statements.

Common mistake

- **"The crowd is always wise."** Independence and $p > 1/2$ are required. In the cascade all six of six people were wrong; with $p = 0.4$ a large jury is almost surely wrong.
- **"Those who follow a cascade are irrational."** Each acted correctly by Bayes; the problem is that information does not accumulate.
- **"Punishment always increases cooperation."** Punishment is costly and gives rise to second-order free riding; antisocial punishment can erase its effect.
- **"A threshold project has a single equilibrium."** Both zero contribution and every profile whose total is exactly 22 are equilibria.
- **"The solution for a common resource is either the state or the market."** Ostrom's local institutions are a third way.
- **"The jury theorem and the Condorcet paradox are the same."** The theorem concerns the probability of finding the truth, the paradox the cycling of preferences.

The five problems below use the tools of the chapter together. Each changes one parameter of an example worked in the text; the solution path is familiar, but the numbers must be recomputed.

Example 12.20

Question. With the prospect theory parameters ($v(x) = x^{0.88}$, $\lambda = 2.25$), is a +300 / −150 coin-toss bet accepted?

Solution. $v(300) = 300^{0.88} \approx 151.3$, $v(-150) = -2.25 \cdot 150^{0.88} \approx -2.25 \cdot 82.2 \approx -185.0$. The prospect value is $\tfrac{1}{2} \cdot 151.3 - \tfrac{1}{2} \cdot 185.0 \approx 75.7 - 92.5 \approx -16.8 < 0$: the bet is rejected. It is rejected although its expected value is +75 because the gain falls below the acceptance threshold ($150 \cdot 2.51 \approx 377$ TL).

Example 12.21

Question. In the commons model ($u_i = g_i(100 - G)$) with $n = 9$ herders, what are the equilibrium G^* and the total payoff? What tax achieves the optimum?

Solution. $g^* = 100/(n + 1) = 100/10 = 10$, $G^* = 90$; the value per animal is 10 and the total is $90 \cdot 10 = 900$ (36% of the optimum). With the tax, $g = (100 - t)/10$; for the optimum $G = 50$, $g = 50/9$, that is, $(100 - t)/10 = 50/9 \Rightarrow 100 - t = 55.6 \Rightarrow t \approx 44.4$.

Example 12.22

Question. In the punished scenario of the Public Goods Game, what would the equilibrium contribution be if the fine were 1 token instead of 2?

Solution. Contributing 3 tokens costs $3 \cdot 0.6 = 1.8$, while the fine is only 1 token. Whatever the others contribute, contributing nothing is $1.8 - 1 = 0.8$ better; contributing 1 or 2 both costs tokens and does not avoid the fine. The equilibrium contribution is 0; a fine rule of this size is ineffective.

Example 12.23

Question. In the Fehr–Schmidt model, at least how many of the 10 tokens does a responder with $\alpha = 2$ demand?

Solution. $s \geq 10\alpha/(1 + 2\alpha) = 20/5 = 4$. Check: for $s = 4$, $4 - 2 \cdot (10 - 8) = 0$; accepting and rejecting give the same utility and the acceptance condition (≥ 0) is met. For $s = 3$, $3 - 2 \cdot 4 = -5 < 0$, reject. The responder demands at least 4 tokens.

Example 12.24

Question. In the Condorcet jury theorem with $p = 0.7$, with what probability is the majority of a 3-person jury right?

Solution. $P_3 = 0.7^3 + 3 \cdot 0.7^2 \cdot 0.3 = 0.343 + 0.441 = 0.784$. A single voter's accuracy of 0.7 rises to 0.784 in a majority of three.

12.8 Summary

- **Bounded rationality**: because information, time, and computation are limited, people look for "good enough." The deviations are systematic, not random; that is why they can be modeled.
- **Prospect theory**: outcomes are evaluated as gains or losses relative to a reference point. The value function is concave for gains, convex for losses, and $\lambda \approx 2.25$ times steeper for losses. The ±100 TL bet is rejected (−36.0); the acceptance threshold is ≈ 251 TL. Probability weighting overstates small probabilities ($w(0.01) \approx 0.055$).
- **The endowment effect and framing** have the same root: when the reference point changes, the same object or the same program is valued differently. For the mug the selling/buying ratio is λ; in the epidemic problem 72% choose the sure option in one frame and 78% the risky option in the other.
- **Quantal response**: in logit choice the better option is chosen more often; $\lambda = 0$ is random, $\lambda \to \infty$ is the best response. In asymmetric Matching Pennies, QRE captures the shift toward the large payoff (0.83 at λ = 0.02) instead of Nash's 1/2.
- **Social preferences**: the average share in the dictator game is about 28%, and offers in the ultimatum game are 40–50%. In the Fehr–Schmidt model a responder with $\alpha = 1$ rejects offers below 10/3; a dictator with $\beta > 1/2$ splits equally. In the trust game 5.16 dollars are sent on average and 4.66 come back: reciprocity responds to intention.
- **The commons**: in the equilibrium with 4 herders $g^* = 20$, total 1,600; the optimum is 2,500 (64%). As n grows the externality grows and the total goes to zero. The Pigouvian tax ($t = 37.5$), quotas, and local institutions based on Ostrom's eight principles are remedies.
- **Welfare criteria**: the utilitarian, egalitarian, and Nash social welfare criteria can choose different policies. Behind the veil of ignorance the Laplace rule leads to the utilitarian criterion (Harsanyi) and maximin to the egalitarian criterion (Rawls).
- **Public goods and punishment**: in theory contributing nothing is dominant; in experiments contributions start at 40–60% and erode. Conditional cooperation and a single free rider produce this decline (10 → 1.3). Punishment protects cooperation but is costly; it can give rise to second-order free riding and antisocial punishment. A punishment imposed as a rule moves the equilibrium from 0 to 3.
- **Threshold projects**: contribution becomes a coordination game; the zero equilibrium and every profile with total exactly 22 are equilibria. Threshold equilibria can be efficient (total 66) but unequal; a refund makes coordination easier.
- **Collective information**: the diversity prediction theorem (the crowd's error = average individual error − diversity); information does not accumulate in a cascade (six of six people are wrong); the jury theorem strengthens the majority for independent and competent voters (5 voters, $p = 0.6 \Rightarrow$

0.683). All three mechanisms rest on independence.

Model	Classical prediction	Observation / the model's prediction
Prospect theory	Indifferent to the ±100 bet	Rejected; threshold ≈ 251 TL
Quantal response	Row plays Top with probability 1/2	Shifts toward the large payoff (0.83)
Fehr–Schmidt	Every positive offer accepted	Offers below 4 rejected when α = 1
The commons	—	Equilibrium 1,600, optimum 2,500
Public goods	0 contribution	Starts at 40–60% and erodes
Punishment	Nobody punishes	Altruistic punishment; contributions stay high
Threshold project	—	The zero equilibrium and the equilibria with total 22
Cascade	—	Two signals determine everyone's guess
Jury theorem	—	If p > 1/2 the majority tends to 1

Table 12.18: The chapter's models at a glance.

12.9 Key terms

bounded rationality satisficing heuristics overconfidence decision under risk prospect theory reference point value function loss aversion probability weighting endowment effect status quo bias framing effect logit choice quantal response equilibrium level-k reasoning dictator game ultimatum game inequity aversion trust game reciprocity collective decision consensus tragedy of the commons externality social optimum Pigouvian tax social welfare utilitarian criterion egalitarian criterion Nash social welfare veil of ignorance Laplace criterion maximin public good public goods game free rider free riding conditional cooperator altruistic punishment second-order free riding antisocial punishment threshold public good wisdom of crowds diversity prediction theorem Bayes' rule information cascade Condorcet jury theorem

12.10 Test Yourself

1. With the prospect theory parameters ($v(x) = x^{0.88}$, $\lambda = 2.25$), is a bet giving +150 TL or −50 TL with equal probability accepted? What is the least gain needed to take the risk of losing 50 TL?

Answer

$v(150) = 150^{0.88} \approx 82.2$, $v(-50) = -2.25 \cdot 50^{0.88} \approx -2.25 \cdot 31.3 \approx -70.4$. The prospect value is $\tfrac{1}{2} \cdot 82.2 - \tfrac{1}{2} \cdot 70.4 \approx 5.9 > 0$: accepted. The threshold is $G = 50 \cdot 2.25^{1/0.88} \approx 50 \cdot 2.51 \approx 125.7$ TL; 150 is above it. The ratio (≈ 2.51) does not depend on the size of the loss.

2. A 10-token ultimatum game is played in the Fehr–Schmidt model and offers are whole numbers. What is the least number of tokens a responder with $\alpha = 0.5$ will accept? What is the answer for a responder with $\alpha = 3$?

   Answer

   The threshold is $10\alpha/(1 + 2\alpha)$. For $\alpha = 0.5$: $5/2 = 2.5$; for $s = 2$, $2 - 0.5 \cdot 6 = -1$ (reject), for $s = 3$, $3 - 0.5 \cdot 4 = +1$ (accept): at least 3. For $\alpha = 3$: $30/7 \approx 4.29$; for $s = 4$, $4 - 3 \cdot 2 = -2$ (reject); $s = 5$ is an equal split and is accepted: at least 5. However large α is, the threshold does not exceed 5.

3. Three herders graze animals on a common pasture; the value per animal is $60 - G$ and herder i's payoff is $g_i(60 - G)$. What are the symmetric Nash equilibrium, the social optimum, the ratio of the equilibrium to the optimum, and the tax per animal that achieves the optimum?

   Answer

   The best response is $g_i = (60 - G_{-i})/2$. Symmetric: $g = (60 - 2g)/2 \Rightarrow 4g = 60 \Rightarrow g^* = 15$, $G^* = 45$; the value per animal is 15, per herder 225, total 675. Check: when the others have 30, $(60 - 30)/2 = 15$. Optimum: $G(60 - G)$ is maximal $\Rightarrow G = 30$, total 900. The ratio is $675/900 = 75\%$. With the tax $g = (60 - t)/4$, $G = 3(60 - t)/4 = 30 \Rightarrow t = 20$.

4. In the Public Goods Game (4 players, 10 tokens each, multiplier 1.6), what would the equilibrium contribution be if the rule were "anyone contributing fewer than 5 tokens is fined 2 tokens"? What would the equilibrium contribution and payoff per player be if the fine were 4 tokens?

   Answer

   Contributing 5 costs $5 \cdot 0.6 = 3$ tokens relative to contributing 0. If the fine is 2, the fine avoided is less than 3: not contributing is better by 1 token, and the equilibrium contribution is 0. If the fine is 4, $4 > 3$: contributing 5 is better by 1 token; more than 5 costs 0.6 per token, and 1–4 both costs tokens and does not avoid the fine. 5 is dominant. Everyone contributes 5: the pool is $20 \cdot 1.6 = 32$, the share per player 8, the payoff $10 - 5 + 8 = 13$.

5. In a threshold project (4 players, 10 tokens each, threshold 22, reward 12), is the profile (9, 9, 4, 0)

a Nash equilibrium? What are the payoffs? What is the answer for the profile (5, 5, 5, 5)?

Answer

(9, 9, 4, 0): the total is 22, the payoffs are (13, 13, 18, 22). If a contributor reduces by 1 the total becomes 21 and the threshold is missed (for example if P3 contributes 3 it gets $7 < 18$); contributing extra is a wasted payment; if P4 contributes it only loses. It is a Nash equilibrium; the total is 66. (5, 5, 5, 5): the total is 20, the threshold is missed and everyone gets 5. If P1 contributes 7 the total becomes 22 and P1 gets $10 - 7 + 12 = 15 > 5$: it is not an equilibrium.

6. In the two-urn cascade model the true urn is A. The signals the five people see, in order, are a, b, b, b, a. What is each person's guess? What would $P(A)$ be if the five signals were pooled?

Answer

Person 1: a ⇒ A, f = 1. Person 2 sees b; together with the earlier guess f = 0, $P(A) = 1/2$; in a tie they follow their own signal ⇒ B, f = 0. Person 3: b ⇒ $P(A) = 1/3$, B, f = −1. Person 4: b ⇒ with their own b f = −2, $P(A) = 1/5$, B; the f inferred from the earlier guesses becomes −2 and a B cascade begins. Person 5 sees a; from the earlier guesses −2, with their own a −1, $P(A) = 1/3 < 1/2$ ⇒ B. The guesses are A, B, B, B, B. Had the five signals been pooled, 2 a's and 3 b's ⇒ f = −1, $P(A) = 1/3$; in this example the signals themselves are misleading, and even pooling the information would not have given the right answer.

7. Each voter independently votes correctly with probability $p = 0.8$. What is the probability that the majority is right in juries of 3 and 5? What happens in a jury of 5 if four members copy the fifth's vote?

Answer

$P_3 = 0.8^3 + 3 \cdot 0.8^2 \cdot 0.2 = 0.512 + 0.384 = 0.896$. $P_5 = 10 \cdot 0.8^3 \cdot 0.2^2 + 5 \cdot 0.8^4 \cdot 0.2 + 0.8^5 = 0.2048 + 0.4096 + 0.32768 \approx 0.942$. If four members copy, the five votes are always the same; the majority is one person's vote and the probability falls to 0.8.

8. In a society with three positions the policies' utilities are $P = (9, 4, 2)$, $Q = (4, 4, 4)$, $R = (6, 5, 3)$. Which policy do the utilitarian criterion (Laplace), the egalitarian criterion (maximin), and Nash social welfare choose?

Answer

The averages are $15/3 = 5$, $12/3 = 4$, $14/3 \approx 4.67$ ⇒ the utilitarian criterion chooses P. The worst positions are 2, 4, 3 ⇒ maximin chooses Q. The products are $9 \cdot 4 \cdot 2 = 72$, $4^3 = 64$, $6 \cdot 5 \cdot 3 = 90$ ⇒ Nash social welfare chooses R. The three criteria choose three different policies.

12.11 Further reading

- D. Kahneman, *Thinking, Fast and Slow* (2011): one of the founders of the theory presents prospect theory, loss aversion, framing, and heuristics in nontechnical language.
- C. F. Camerer, *Behavioral Game Theory: Experiments in Strategic Interaction* (2003): a comprehensive survey of the experimental findings of behavioral game theory, from dictator, ultimatum, and trust games to quantal response and level-k models.
- E. Ostrom, *Governing the Commons* (1990): gives the case studies of local governance of common resources (including the Alanya fishermen) and the emergence of the design principles in detail.
- D. Easley and J. Kleinberg, *Networks, Crowds, and Markets* (2010), Chapter 16: presents information cascades with the urn model of this chapter, showing the Bayesian calculation step by step.

12.12 References

1. L. R. Anderson, C. A. Holt, "Information cascades in the laboratory", *American Economic Review* 87(5), 1997.
2. D. Austen-Smith, J. S. Banks, "Information aggregation, rationality, and the Condorcet jury theorem", *American Political Science Review* 90(1), 1996.
3. J. Berg, J. Dickhaut, K. McCabe, "Trust, reciprocity, and social history", *Games and Economic Behavior* 10(1), 1995.
4. S. Bikhchandani, D. Hirshleifer, I. Welch, "A theory of fads, fashion, custom, and cultural change as informational cascades", *J. Political Economy* 100(5), 1992.
5. Marquis de Condorcet, *Essai sur l'application de l'analyse à la probabilité des décisions rendues à la pluralité des voix*, 1785.
6. C. Engel, "Dictator games: A meta study", *Experimental Economics* 14(4), 2011.
7. T. Feddersen, W. Pesendorfer, "Convicting the innocent", *American Political Science Review* 92(1), 1998.
8. E. Fehr, S. Gächter, "Cooperation and punishment in public goods experiments", *American Economic Review* 90(4), 2000.
9. E. Fehr, S. Gächter, "Altruistic punishment in humans", *Nature* 415, 2002.
10. E. Fehr, K. M. Schmidt, "A theory of fairness, competition, and cooperation", *Quarterly J. Economics* 114(3), 1999.
11. U. Fischbacher, S. Gächter, E. Fehr, "Are people conditionally cooperative? Evidence from a public goods experiment", *Economics Letters* 71(3), 2001.
12. F. Galton, "Vox populi", *Nature* 75, 1907.
13. J. K. Goeree, C. A. Holt, "Ten little treasures of game theory and ten intuitive contradictions", *American Economic Review* 91(5), 2001.
14. G. Hardin, "The tragedy of the commons", *Science* 162, 1968.

15. J. C. Harsanyi, “Cardinal welfare, individualistic ethics, and interpersonal comparisons of utility”, *J. Political Economy* 63, 1955.

16. B. Herrmann, C. Thöni, S. Gächter, “Antisocial punishment across societies”, *Science* 319, 2008.

17. D. Kahneman, J. L. Knetsch, R. H. Thaler, “Experimental tests of the endowment effect and the Coase theorem”, *J. Political Economy* 98(6), 1990.

18. D. Kahneman, A. Tversky, “Prospect theory: An analysis of decision under risk”, *Econometrica* 47(2), 1979.

19. J. O. Ledyard, “Public goods: A survey of experimental research”, J. H. Kagel, A. E. Roth (ed.), *The Handbook of Experimental Economics*, Princeton University Press, 1995.

20. R. D. McKelvey, T. R. Palfrey, “Quantal response equilibria for normal form games”, *Games and Economic Behavior* 10(1), 1995.

21. R. Nagel, “Unraveling in guessing games: An experimental study”, *American Economic Review* 85(5), 1995.

22. E. Ostrom, *Governing the Commons*, Cambridge University Press, 1990.

23. S. E. Page, *The Difference*, Princeton University Press, 2007.

24. M. Rabin, “Incorporating fairness into game theory and economics”, *American Economic Review* 83(5), 1993.

25. J. Rawls, *A Theory of Justice*, Harvard University Press, 1971.

26. H. A. Simon, “A behavioral model of rational choice”, *Quarterly J. Economics* 69, 1955.

27. A. Tversky, D. Kahneman, “The framing of decisions and the psychology of choice”, *Science* 211, 1981.

28. A. Tversky, D. Kahneman, “Advances in prospect theory: Cumulative representation of uncertainty”, *J. Risk and Uncertainty* 5, 1992.

PART V

Dynamics and the Future

Chapters 13–14

PART V · CHAPTER 13

Learning and Adaptation

The classical question of game theory is "what happens in equilibrium?" It assumes that the players know the game and can compute the equilibrium. In practice, players often do not know the payoff table and lack the computing power to find an equilibrium; they simply play, observe the outcome, and adjust their behavior a little the next time. This chapter studies where such simple update rules lead in the long run. Best-response dynamics and fictitious play look at the opponent's history; the Hedge algorithm minimizes regret; multi-armed bandit algorithms balance exploration against exploitation; Q-learning propagates reward backward through states; replicator dynamics models the spread of successful strategies in a population; and the repeated Prisoner's Dilemma shows how cooperation can be learned and sustained.

LEARNING OUTCOMES

Dynamics
Run best-response dynamics and fictitious play step by step, and explain when they converge to an equilibrium.

Regret
Compute external regret and interpret the no-regret property of the multiplicative weights (Hedge) algorithm.

Exploration–exploitation
Trace and compare the ε-greedy and UCB algorithms on the multi-armed bandit problem.

Evolution and repetition
Explain replicator dynamics and the success of Tit-for-Tat in the repeated Prisoner's Dilemma.

13.1 Introduction

Equilibrium analysis usually solves games "from the outside": the payoff matrix is given, the equilibrium is computed, and the players are assumed to play it. In this chapter the viewpoint changes. Each player is an algorithm. In every round it picks an action, observes the outcome, and

updates its rule. The question is no longer "what is the equilibrium?" but "where does this update rule lead in the long run?"

13.1.1 Opening game: the multi-armed bandit

The chapter's opening game is a slot machine. The machine has three arms. Each arm pays 1 point with a hidden probability and 0 otherwise. Pulling an arm reveals only that arm's outcome; what the other arms would have paid in that round stays unknown. The goal is to maximize the total reward over a fixed number of pulls. A slot machine with a single lever is called a "one-armed bandit," so the problem with several levers is called the **multi-armed bandit** problem.

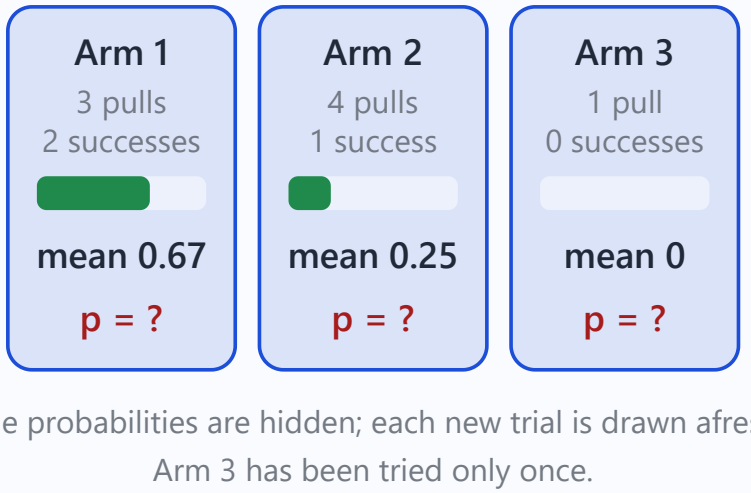


Figure 13.1: The opening situation. Arm 1 has been pulled three times with two successes (mean 0.67); Arm 2 has one success in four pulls (0.25); Arm 3 has been pulled once and failed (0). The true probabilities are hidden.

Figure 13.1 shows the situation after eight pulls. Arm 1 has the highest observed mean, $2/3 \approx 0.67$. Arm 3 failed on its only trial. Two reasonable thoughts collide. The first says that Arm 1 looks best and the remaining pulls should go to it. The second says that almost nothing is known about Arm 3: even if its true probability is high, a single failure is not surprising. Trying Arm 3 again yields information, but that pull means giving up the current best guess.

This tension is called the **exploration–exploitation dilemma**. **Exploration** means trying little-known options to gather information. **Exploitation** means using the option that looks best so far. A player who only exploits may get stuck on the wrong arm; a player who only explores never uses what it has learned. For example, in a game of one hundred pulls there is no single right answer to how many pulls should be spent on exploring; the answer depends on how different the arms are and how long the game lasts. Two algorithms (ε-greedy and UCB1) give numerical answers to this question; the opening situation is resolved in 13.4.5.

13.1.2 Why learning and adaptation?

First, let us separate two terms. **Learning** means developing a strategy from past experience. **Adaptation** means updating the strategy when the opponent or the environment changes. The two often go together: a bidding algorithm learns from yesterday's auctions and adapts again today when the competitors change.

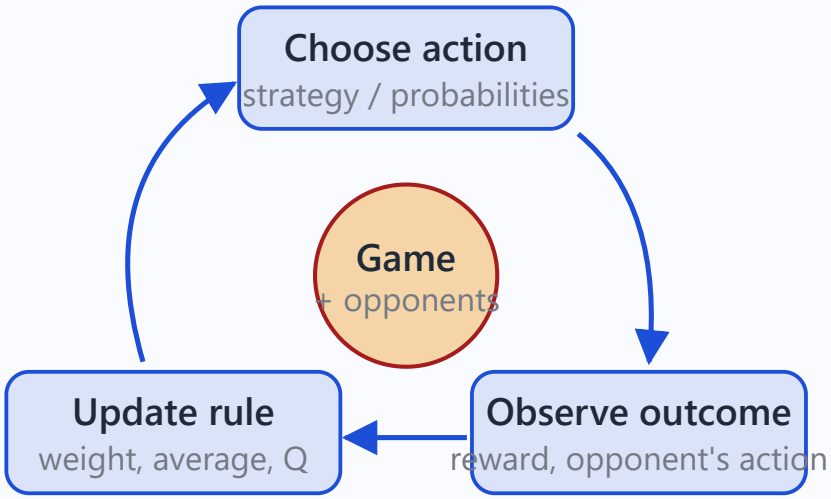


Figure 13.2: The learning loop. The player chooses an action (or a probability distribution), the game and the opponents produce an outcome, and the player observes it and updates its rule (a weight, an average, or a Q-value).

Figure 13.2 shows the skeleton common to all algorithms in this chapter. They differ at only two points: what they observe and how they use the observation. Three observations explain why learning models are needed.

1. **Computing an equilibrium can be hard.** Finding a mixed Nash equilibrium in a general game is a PPAD-complete problem. PPAD-complete is a class of problems for which no fast (polynomial-time) method is known on large games. If even computers cannot find an equilibrium quickly, it is hard to assume that players compute one in their heads. How, then, do players "find" equilibria?
2. **Players may not know the game.** The payoff matrix is often unknown. An advertiser does not see its competitors' valuations; a driver does not see other drivers' route choices. The player only plays and sees the result.
3. **The real question is the power of simple rules.** Do simple, local update rules lead to equilibrium in the long run, or at least to good performance? The answer depends on the rule and the game: some rules reach equilibrium in some games, some cycle, and some are "no-regret" even when they do not reach equilibrium.

The examples are familiar from daily life. Chess and poker AIs learn by playing millions of games against themselves. Bidding algorithms in ad auctions look at the results every day and adjust their bids. Navigation apps change routes according to traffic; since each driver's route affects the other

drivers' traffic, this too is a game.

13.1.3 Overview of learning models

Six learning models are studied in this chapter. Table 13.1 compares them along two questions: what does the player observe, and how does it update?

Model	What does the player observe?	How does it update?
Best-response dynamics	The opponent's last action	Plays the best response to it
Fictitious play	The opponent's entire history	Best response to the frequencies
Regret minimization (Hedge)	The payoff of every action	Multiplicatively increases the weight of good actions
Multi-armed bandit (ε-greedy, UCB)	Only the reward of the chosen action	Average estimate + exploration
Reinforcement learning (Q-learning)	Reward and new state	Updates the $Q(s, a)$ values
Replicator dynamics	Average success in the population	Successful strategies spread

Table 13.1: Learning models: the information the player observes and the update rule.

Reading the table from top to bottom reveals a trend. In the first two models the player watches the opponent: it knows what the opponent played and computes the best response accordingly. This requires the player to know its own payoff table. In Hedge the player watches not the opponent but the payoffs of its own actions; it does not need to know what the opponent did. In bandits and Q-learning the information shrinks further: only the reward of the chosen action is seen. In the last row the viewpoint moves from the individual to the population; no single player "thinks," and successful behavior simply multiplies.

Another difference among the models is between **full feedback** and **partial feedback**. With full feedback the player also learns what the actions it did not play would have paid in that round. For example, whichever stock an investor buys, in the evening the returns of all stocks are visible. With partial feedback only the outcome of the chosen action is seen. For example, a physician who gives a patient one drug never learns what the other drug would have done for that patient. Hedge is designed for full feedback and bandit algorithms for partial feedback; the need for exploration arises from this difference.

> **Common mistake**
>
> Equating the word "learning" with artificial neural networks. Most algorithms in this chapter are a few lines of counters and weight updates. What defines learning is not complexity but that behavior changes according to observed outcomes.

13.2 Best response and fictitious play

The simplest learning rules follow directly from the definition of Nash equilibrium. A Nash equilibrium is a profile in which every player best-responds to the others' choices. So if players take turns best-responding, at some point they may reach a profile in which everyone is best-responding, that is, an equilibrium. This section examines when the idea works, when it falls into a cycle, and how the cycle can be broken by remembering the past.

A reminder of some background. Player i's **best response** to the others' choice s_{-i} is the strategy that maximizes its payoff: $BR_i(s_{-i}) = \operatorname{argmax}_{s_i} u_i(s_i, s_{-i})$. In a two-player matrix game, the row player's best response is the row that gives the largest first number in the opponent's column; the column player's best response is the column that gives the largest second number in the opponent's row. A profile is a Nash equilibrium if both players best-respond to each other.

13.2.1 Best-response dynamics step by step

Definition 13.1

Best-response dynamics: start from an initial profile. The players take turns changing their own action so as to best-respond to the opponents' *current* action. The dynamics stop when no player wants to change its action.

At the moment the dynamics stop, every player is best-responding to the others' actions; this is the definition of a Nash equilibrium. Hence if best-response dynamics *stop*, they stop at a pure Nash equilibrium. The real question is whether the dynamics stop at all. First, let us look at an example where they do.

	X	Y	Z
A	**(4, 3)**	(0, 2)	(1, 1)
B	(2, 1)	**(3, 3)**	(0, 2)
C	(1, 4)	(2, 0)	(3, 2)

Table 13.2: A three-by-three example game. In a cell (u_1, u_2): the row player's payoff first, then the column player's. The green cells are the two pure Nash equilibria.

Example 13.1

In the game of Table 13.2, let the dynamics start from the profile (B, Z). Let the row player move first, then the column player.

1. **Row player against Z:** in column Z the first numbers are A: 1, B: 0, C: 3. The largest is 3, so the row player switches to C. The profile becomes (C, Z).
2. **Column player against C:** in row C the second numbers are X: 4, Y: 0, Z: 2. The largest is 4, so the column player switches to X. The profile becomes (C, X).
3. **Row player against X:** in column X the first numbers are A: 4, B: 2, C: 1. The row player switches to A. The profile becomes (A, X).
4. **Column player against A:** in row A the second numbers are X: 3, Y: 2, Z: 1. The column player is already choosing X (3 > 2, 1); it does not change.
5. **Row player against X again:** A is already the best (4 > 2, 1). Nobody wants to change; the dynamics stop at (A, X), and (A, X) is a Nash equilibrium.

Had the same dynamics started from (C, Y), the outcome would have been different. Against Y the row player sees A: 0, B: 3, C: 2 and switches to B. Against B the column player sees X: 1, Y: 3, Z: 2 and stays at Y. The dynamics stop at the equilibrium (B, Y).

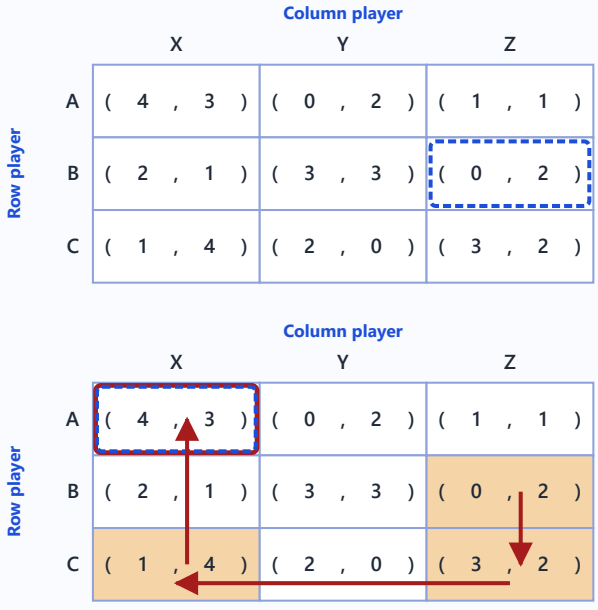


Figure 13.3: Best-response dynamics. On the left, the starting profile (B, Z) is shown with a dashed frame. On the right, the path the dynamics follow: (B, Z) → (C, Z) → (C, X) → (A, X). The orange cells are the profiles on the path; the dynamics stop at the Nash equilibrium (A, X).

In Figure 13.3 each arrow is a change by a single player. Vertical arrows are moves of the row player (only the row changes, the column stays fixed), and horizontal arrows are moves of the column player. The example yields two important observations. First, the dynamics stopped at an equilibrium; at each step it sufficed for one player to do a local computation, and nobody "solved" for the equilibrium. Second, *which* equilibrium is reached depends on the starting point. In games with several equilibria the learning rule acts like an equilibrium selection mechanism: dynamics that start at (B, Z) settle at (A, X), and dynamics that start at (C, Y) settle at (B, Y).

Common mistake

- **Updating both players at the same time:** in this example the players move in turn. Had both best-responded simultaneously, a different path would have been followed; for example, at (B, Z) the row player moves to C and the column player moves to Y, the best response to B, at the same time, and (C, Y) is reached.
- **Looking at the wrong number:** the row player always wants to maximize the first number and the column player the second. Looking at the row player's payoff when computing the column player's move is the most common error.

13.2.2 When do best-response dynamics converge?

The dynamics do not stop in every game. The simplest counterexample is the **Matching Pennies** game. Two players simultaneously show one face of a coin, Heads (H) or Tails (T). If the faces are the same, the *Matcher* wins (gets 1, the opponent −1); if they differ, the *Mismatcher* wins.

Match. \ Mism.	H	T
H	(1, −1)	(−1, 1)
T	(−1, 1)	(1, −1)

Table 13.3: Matching Pennies. Row: Matcher, column: Mismatcher. A cell gives the payoffs (Matcher, Mismatcher).

Example 13.2

1. At (H, H) the Matcher is winning and is satisfied. The Mismatcher wants a different face and switches to T: (H, T).
2. At (H, T) the Mismatcher is satisfied. The Matcher wants the same face and switches to T: (T, T).
3. At (T, T) the Mismatcher switches to H: (T, H).
4. At (T, H) the Matcher switches to H: (H, H) again. The cycle starts over.

The sequence (H, H) → (H, T) → (T, T) → (T, H) → (H, H) → … goes on forever.

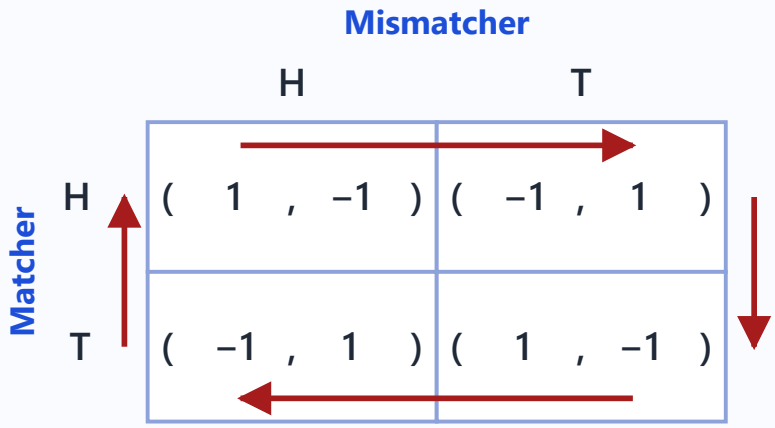


Figure 13.4: Best-response dynamics in Matching Pennies. The arrows form a clockwise cycle among the four profiles; the dynamics never stop at any cell.

As Figure 13.4 shows, at every step one player is happy and the other is unhappy. The game has no pure Nash equilibrium; in each of the four cells the losing player wants to deviate. So the dynamics cannot stop anywhere. The game's only equilibrium is mixed: both players play H and T with probability ½. Best-response dynamics, however, only jump between pure actions; they never produce "fifty-fifty" behavior, so they cannot reach the mixed equilibrium either.

The question of in which games the dynamics are guaranteed to stop has a well-known answer: **potential games**.

Definition 13.2

A game is a **potential game** if a single function Φ common to all players can be found such that whenever any one player changes its action alone, the change in its own payoff equals the change in Φ.

$$u_i(s_i', s_{-i}) - u_i(s_i, s_{-i}) = \Phi(s_i', s_{-i}) - \Phi(s_i, s_{-i})$$

The potential function gathers all the players' selfish calculations into a single "common scoreboard." A player's profitable deviation raises the common score by the same amount. Congestion games, that is, traffic games in which each driver chooses a road and a road's cost depends on the number of drivers using it, are the most important examples of potential games (Monderer and Shapley, 1996).

Theorem 13.1 (Convergence in potential games)

In a finite potential game, best-response dynamics (and, more generally, any sequence of unilateral changes that strictly increase the mover's payoff at every step) end after finitely many steps; if the sequence is continued until no profitable deviation remains, it stops at a pure Nash equilibrium.

Proof. Every step of the dynamics is a *profitable* deviation: the moving player's payoff strictly increases. By the definition of the potential, this increase equals the increase in Φ; so Φ strictly increases at every step. Since the game is finite, the number of strategy profiles is finite, and therefore the set of values Φ can take is finite. A strictly increasing sequence cannot continue forever in a finite set; nor can it return to the same profile twice, because on returning Φ would take its old value again. Hence the dynamics stop after finitely many steps. At the profile where they stop, no player has a profitable deviation; this profile is a pure Nash equilibrium. □

A corollary follows from the proof: Matching Pennies cannot be a potential game. If it were, each of the four steps in the cycle would increase Φ, and after four steps, when the same profile is reached again, Φ would be larger than its starting value; this is a contradiction. A dynamics that enters a cycle is proof that no potential exists.

Example 13.3

Two friends want to use the same software. If both choose A, each gets 2; if both choose B, each gets 1; if they choose differently, they get 0. This is a coordination game, and its potential can be chosen as $\Phi(A, A) = 2$, $\Phi(B, B) = 1$, $\Phi(A, B) = \Phi(B, A) = 0$: in every unilateral change, the payoff difference of the player who changes equals the difference in Φ.

1. Start at (A, B), $\Phi = 0$. Against B the row player gets 0 with A and 1 with B, and switches to B: (B, B), $\Phi = 1$.
2. At (B, B) the column player gets 1 with B and 0 with A; it does not change. The row player does not change either. The dynamics stop at the equilibrium (B, B).

Φ rose from 0 to 1 and stayed there. The dynamics did not go to the equilibrium (A, A) with the higher potential: the convergence guarantee says that some equilibrium will be reached, not that the best equilibrium will be.

For games with no pure equilibrium, the remedy is to respond not only to the last action but to the opponent's entire history. This idea gives rise to fictitious play.

13.2.3 Fictitious play: Matching Pennies

Definition 13.3

Fictitious play: in every round, a player counts the frequencies of the opponent's actions so far and "pretends" that the opponent is using a mixed strategy that plays with these frequencies. It then plays the best response to this mixed strategy.

The "fiction" in the name is the simplifying assumption the player makes about the opponent: the opponent is treated as if it were playing with a fixed die whose probabilities equal the past frequencies. For example, if the opponent has played T in five of seven rounds, it is assumed to play $P(H) = 2/7$, $P(T) = 5/7$. The difference from best-response dynamics is that the whole history is used instead of a single last observation. For this reason the belief in fictitious play changes slowly: a single new observation carries a small weight within a long history.

In Matching Pennies this rule takes a very simple form. Since the Matcher wants the same face, it plays whichever face is the majority in the opponent's history. Since the Mismatcher wants a different face, it plays the opposite of the majority in the opponent's history. In case of a tie, both play H. In the first round, with no history, both play H arbitrarily.

Example 13.4

That the rule "match the majority" is a best response can be verified with expected payoffs. In round 8 the Matcher sees that the Mismatcher played H twice and T five times in the previous seven rounds; its belief is $P(H) = 2/7$.

1. If the Matcher plays H: $2/7 \cdot 1 + 5/7 \cdot (-1) = -3/7$.
2. If the Matcher plays T: $2/7 \cdot (-1) + 5/7 \cdot 1 = 3/7$.
3. Since $3/7 > -3/7$, the best response is T; T is the majority in the opponent's history.

In general, if the belief is $P(H) = q$, the expected payoff of H is $2q - 1$ and that of T is $1 - 2q$. If $q > 1/2$, H is better; if $q < 1/2$, T is better; at $q = 1/2$ the two are equal and the rule chooses H.

Applying the rule for ten rounds gives Table 13.4. In the table the "history" columns count the rounds *before* that round, while the "rate" columns include that round.

t	Mism. history H/T	Matcher	Match. history H/T	Mismatcher	Match. H rate	Mism. H rate
1	0 / 0	H	0 / 0	H	1.00	1.00
2	1 / 0	H	1 / 0	T	1.00	0.50
3	1 / 1	H	2 / 0	T	1.00	0.33
4	1 / 2	T	3 / 0	T	0.75	0.25

t	Mism. history H/T	Matcher	Match. history H/T	Mismatcher	Match. H rate	Mism. H rate
5	1 / 3	T	3 / 1	T	0.60	0.20
6	1 / 4	T	3 / 2	T	0.50	0.17
7	1 / 5	T	3 / 3	H	0.43	0.29
8	2 / 5	T	3 / 4	H	0.38	0.38
9	3 / 5	T	3 / 5	H	0.33	0.44
10	4 / 5	T	3 / 6	H	0.30	0.50

Table 13.4: Fictitious play, Matching Pennies, ten rounds. "Mism. history" gives the H/T counts the Matcher sees, and "Match. history" the counts the Mismatcher sees (earlier rounds). The rates are the fractions of H played up to and including that round.

The first few rows of the table are worth reading in detail.

- **t = 1:** There is no history; both players play H arbitrarily.
- **t = 2:** The Matcher sees 1 / 0 in the opponent's history; the majority is H, so it plays H. The Mismatcher also sees 1 / 0 in the opponent's history and plays T, the opposite of the majority.
- **t = 3:** The Matcher sees 1 / 1 and plays H on the tie. The Mismatcher sees 2 / 0 and plays T.
- **t = 4:** The Matcher sees 1 / 2; the majority is now T, and the Matcher switches to T. The Mismatcher sees 3 / 0 and stays at T. The Matcher's H rate drops to *3/4 = 0.75*.
- **t = 7:** The Mismatcher sees 3 / 3 and switches to H on the tie. It thus answers the Matcher's shift to T with a delay of three rounds.

The actions still cycle: first (H, H), then (H, T), (T, T) and (T, H). There is, however, an important difference from best-response dynamics. Each phase lasts several rounds, because for the belief to reverse, the majority in the past must change. The rates do not jump; they change gradually: at the end of ten rounds the Matcher's H rate is 0.30 and the Mismatcher's is 0.50.

Common mistake

- **Looking at one's own history:** in fictitious play a player counts the *opponent's* history. The Matcher does not decide by its own count of H.
- **Including the current round in the history:** the decision in round t is based on the rounds before round t. The action of the current round is not yet known.
- **Thinking that the actions converge to equilibrium:** what approaches the equilibrium is the *frequencies*; the action played in each round is still a pure H or T.

13.2.4 Code trace: fictitious()

The following function runs fictitious play in Matching Pennies for *T* rounds. Actions are encoded by numbers: 0 = H, 1 = T.

```
def fictitious(T):
    nm, nx = [0, 0], [0, 0]   # H/T counters
    a, b = 0, 0               # round 1: both H
    for t in range(1, T + 1):
        nm[a] += 1            # matcher's history
        nx[b] += 1            # mismatcher's history
        # matcher: match the mismatcher's majority
        a = 0 if nx[0] >= nx[1] else 1
        # mismatcher: avoid the matcher's majority
        b = 1 if nm[0] > nm[1] else 0
    return nm, nx
```

The parts of the code are as follows.

- `nm` and `nx` are counter lists. `nm[0]` holds how many times the Matcher has played H so far and `nm[1]` how many times it has played T; `nx` holds the same information for the Mismatcher.
- `a` and `b` are the actions to be played in this round; in the first round both are initialized to 0 (H).
- At the start of the loop the actions of this round are added to the counters; that is, the round has been played.
- Next, the actions of the following round are computed. The Matcher plays H if the Mismatcher's H count is not smaller than its T count (including ties): `a = 0 if nx[0] >= nx[1] else 1`.
- The Mismatcher plays T if the Matcher's H count is strictly greater than its T count, and H otherwise: `b = 1 if nm[0] > nm[1] else 0`. On a tie the condition is false and H is chosen; this is the "H on ties" rule of the table.

Let the call be `fictitious(6)`. Table 13.5 shows the state of the counters at the end of each round and the decision for the next round.

Round	Played (a, b)	Winner	nm	nx	Next round
1	(H, H)	Matcher	1 / 0	1 / 0	(H, T)
2	(H, T)	Mismatcher	2 / 0	1 / 1	(H, T)
3	(H, T)	Mismatcher	3 / 0	1 / 2	(T, T)
4	(T, T)	Matcher	3 / 1	1 / 3	(T, T)
5	(T, T)	Matcher	3 / 2	1 / 4	(T, T)
6	(T, T)	Matcher	3 / 3	1 / 5	—

Table 13.5: Trace of `fictitious(6)`. The counters are the values after the round has been played (H/T).

The critical moments of the trace are rounds 2 and 3. At the end of round 2 the Mismatcher's counters are equal, 1 / 1; the Matcher keeps playing H on the tie. At the end of round 3 the Mismatcher's counters become 1 / 2 and the Matcher switches to T. The Mismatcher, on the other hand, sees an H

majority in the Matcher's history until the end of round 6 and keeps playing T. The function returns `nm = [3, 3]`, `nx = [1, 5]`. The H rates are *3/6 = 0.50* and *1/6 ≈ 0.17*; these agree with row 6 of Table 13.4.

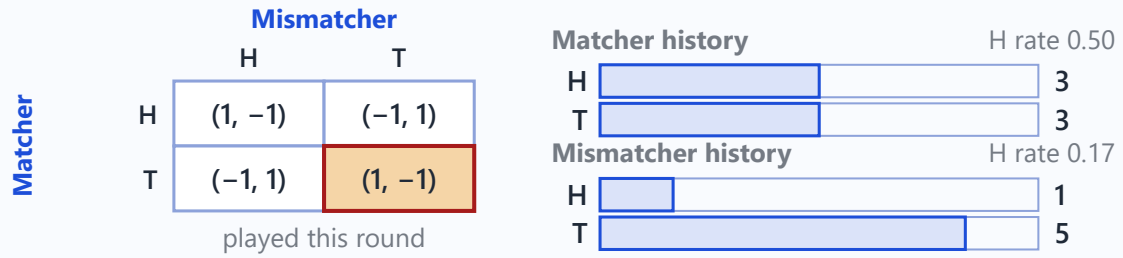


Figure 13.5: The end of the call `fictitious(6)`. On the left, the cell (T, T) played in round 6; on the right, the two players' H/T counters: the Matcher 3 / 3 (H rate 0.50), the Mismatcher 1 / 5 (H rate 0.17).

13.2.5 The long run: frequencies converge to equilibrium

When fictitious play is continued for four hundred rounds, the actions keep changing in cycles: (H, H) → (H, T) → (T, T) → (T, H) → (H, H) … But each cycle is longer than the previous one. As the history grows, more rounds are needed to reverse the majority. The frequencies, however, approach the mixed equilibrium, that is, ½: in four hundred rounds the Matcher's H rate is 0.472 and the Mismatcher's is 0.492.

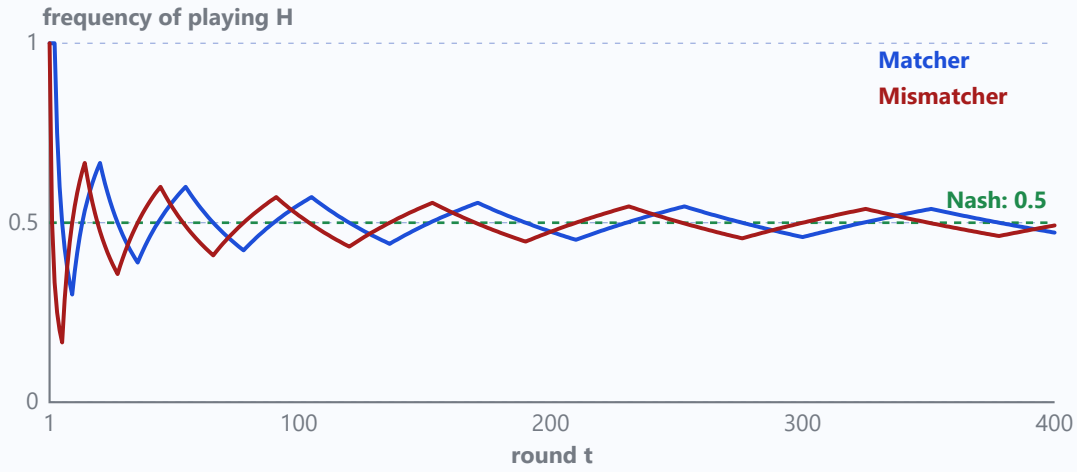


Figure 13.6: The frequency of playing H over four hundred rounds of fictitious play. Blue: Matcher, red: Mismatcher. The amplitude of the oscillations shrinks and both curves approach the Nash equilibrium value 0.5.

In Figure 13.6 one can sense why the oscillations die out. When a player's frequency moves away from ½, the opponent's best response turns in the direction that punishes it, and the deviation is corrected after a while. For example, while the Matcher's H rate is above ½, the Mismatcher plays T;

the Matcher eventually switches to T and lowers its own H rate. As the history grows, the effect of a single round on the frequency shrinks to $1/t$, so the amplitude of the oscillation shrinks as well.

This behavior is not seen in every game. The convergence of frequencies to equilibrium in fictitious play has been proved for the following classes of games:

- **Two-player zero-sum games** (Robinson, 1951). Matching Pennies belongs to this class.
- **2×2 games**, that is, games in which each of the two players has two strategies (Miyasawa, 1961).
- **Potential games** (Monderer and Shapley, 1996).

On the other hand, Shapley (1964) constructed a general-sum 3×3 game. In this game the actions go around in ever longer cycles and the frequencies never approach any equilibrium. Shapley's example is the classic result showing that the expectation "learning leads to equilibrium in every game" is wrong.

> **Common mistake**
>
> Concluding that "the frequencies converge to equilibrium, so the players are playing the equilibrium." The actions played in each round are pure and the cycle goes on; only the long-run average approaches (½, ½). An observer who looks only at the last rounds sees not the equilibrium but a long run of H or T.

13.3 Regret and Hedge

Best-response dynamics and fictitious play assume that the player knows the payoff table and observes the opponent's actions. Here this assumption is weakened. The player does not model the opponent; after each round it only sees what its own actions earned (or would have earned). The opponent may be a human, an algorithm, or a malicious adversary. Under these conditions, what does it mean to "play well"? The answer is the concept of regret.

13.3.1 External regret

In everyday language, regret means looking back and saying "I wish I had done that." The definition in learning theory is the numerical version of this feeling. After the game is over, one looks back and asks: had the same action been played from start to finish, what would the best case have earned? The difference between this number and what was actually earned is the regret.

Definition 13.4

Over T rounds, let the payoff of action i in round t be $g_t(i)$ and the action chosen by the algorithm be a_t. The **external regret** is the difference between the total of the best fixed action in hindsight and the algorithm's total:

$$R_T = \max_i \sum_{t=1}^{T} g_t(i) - \sum_{t=1}^{T} g_t(a_t)$$

If the algorithm chooses randomly, the second sum is computed with expected payoffs. If $R_T/T \to 0$ as T grows, the algorithm is **no-regret**.

Three details of the definition matter. First, the comparison is made with a *fixed* action, not with an "oracle" that picks the best action in every round after the fact. Comparing with the oracle would be a race that no algorithm could win. Second, the best fixed action is known only after the game is over; the algorithm does not know it during the game. Third, no-regret is an average-case measure. R_T may grow, but it must grow more slowly than T. For example, if $R_T = 2\sqrt{T}$ then $R_T/T = 2/\sqrt{T}$ goes to zero; the per-round loss vanishes over time. If $R_T = T/2$, the per-round loss is stuck at ½.

Example 13.5

In a three-round game with two actions, let the payoffs be $g_1 = (1, 0)$, $g_2 = (0, 1)$, $g_3 = (1, 0)$. Let an algorithm play B, A, A in that order.

1. The algorithm's total: 0 with B in round 1, 0 with A in round 2, 1 with A in round 3. Total 1.
2. Totals of the fixed actions: always A gives $1 + 0 + 1 = 2$, always B gives $0 + 1 + 0 = 1$. The best is 2.
3. The external regret is $R_3 = 2 - 1 = 1$.

An oracle that picks the best action in every round would have earned 3; but the regret definition uses not this 3 but the 2 of the best fixed action.

13.3.2 Why does "follow the leader" fail?

The first rule that comes to mind for reducing regret is this: play the action with the largest total payoff so far. This rule is called **follow the leader**. It closely resembles fictitious play; it is the best response to the past. Unfortunately, against a badly chosen payoff sequence this rule is a disaster. Table 13.6 is an example; on ties A is chosen.

t	Previous total A / B	g(A)	g(B)	Follow the leader	Payoff
1	0.0 / 0.0	0.5	0.0	A	0.5
2	0.5 / 0.0	0.0	1.0	A	0.0

t	Previous total A / B	g(A)	g(B)	Follow the leader	Payoff
3	0.5 / 1.0	1.0	0.0	B	0.0
4	1.5 / 1.0	0.0	1.0	A	0.0
5	1.5 / 2.0	1.0	0.0	B	0.0
6	2.5 / 2.0	0.0	1.0	A	0.0

Table 13.6: Follow the leader, two actions, six rounds. "Previous total" is the accumulated payoff before that round.

Example 13.6

1. **t = 1:** The totals are equal (0 / 0); A is chosen on the tie. A pays 0.5 in that round. The algorithm's total is 0.5.
2. **t = 2:** The leader is A (0.5 > 0). But in this round A pays 0 and B pays 1; the algorithm gets 0.
3. **t = 3:** The totals are 0.5 / 1.0; the leader is B. In this round B pays 0 and A pays 1; the algorithm again gets 0.
4. **t = 4, 5, 6:** The same pattern continues. The leader is exactly the loser of the round every time; the algorithm's total stays at 0.5.
5. **Fixed actions:** always A gives $0.5 + 0 + 1 + 0 + 1 + 0 = 2.5$; always B gives $0 + 1 + 0 + 1 + 0 + 1 = 3$. The best fixed action is B.
6. **Regret:** $R_6 = 3 - 0.5 = 2.5$, that is, about $T/2$.

The trick of the sequence is the small 0.5 difference in the first round. This difference keeps the leader "one step behind" in every round: since the payoffs alternate between 1 and 0, the action that pulled ahead in the previous round pays 0 in this round. As the sequence is extended, the algorithm's total stays at 0.5 while the total of the best fixed action becomes about $T/2$. The regret grows linearly: $R_T/T \to 1/2$, it does not go to zero. Follow the leader is not no-regret.

The source of the problem is that the rule is **deterministic**. An adversary who knows the rule can compute in advance which action the algorithm will choose and give that action payoff 0 and the other 1. Then the algorithm gets 0 every round. Since the total payoff of the two actions is 1 in every round, over T rounds the two actions' totals add up to T, and the larger of them is at least $T/2$. So *every* deterministic algorithm has regret at least $T/2$ in the worst case. The only way to be no-regret is to play randomly: even if the adversary knows the probabilities, it cannot know which action will come up.

Common mistake

The assumption that "what was best in the past will be best in the future." If payoffs come from a fixed distribution, follow the leader works well. In games, however, the opponents determine the payoffs. If an adversary chooses the payoffs, trusting the past completely is an

invitation to be exploited.

13.3.3 Multiplicative weights (Hedge)

Follow the leader has two weaknesses: it chooses with certainty, and it fully trusts a small difference. The **multiplicative weights** algorithm, better known as **Hedge**, removes both weaknesses at once. The leader is chosen with high probability but not with certainty; if the difference is small, the probabilities stay close to each other.

Definition 13.5

The **Hedge** algorithm assigns a weight to each of n actions and uses a parameter $\eta > 0$, the **learning rate**:

1. Initially every action gets the same weight: $w_i = 1$.
2. In every round, action i is played with probability $p_i = w_i / \Sigma_j w_j$.
3. At the end of the round the payoffs of all actions are observed and the weights are updated: $w_i \leftarrow w_i \cdot e^{\eta \cdot g_t(i)}$.

The update rule is multiplicative: the weight of an action with payoff 1 is multiplied by e^{η}, and the weight of an action with payoff 0 does not change. For example, if $\eta = \ln 2$ is chosen, then $e^{\eta} = 2$ and the weight of an action with payoff 1 exactly doubles. For $\eta = 0.5$ the factor is $e^{0.5} \approx 1.65$.

When the updates are applied one after another, the weight becomes an exponential function of the action's total payoff. If G_i is the total payoff of action i so far, then $w_i = e^{\eta \cdot G_i}$, because $e^{\eta \cdot g_1} \cdot e^{\eta \cdot g_2} \cdot \ldots = e^{\eta \cdot (g_1 + g_2 + \ldots)}$. This form makes it clear that Hedge is a "soft" follow the leader. The leader has the largest G_i, so it receives the largest probability. But if the difference between the totals of two actions is Δ, the ratio of their probabilities is $e^{\eta \cdot \Delta}$; a small difference means a small advantage. If η grows very large, the ratio goes to infinity and Hedge turns into follow the leader; if η approaches zero, all probabilities become equal and Hedge turns into random play.

Theorem 13.2 (Regret bound of Hedge)

If the payoffs lie in the interval $[0, 1]$ and there are n actions, the external regret of Hedge is bounded by

$$R_T \le (\ln n)/\eta + \eta\, T$$

If $\eta = \sqrt{(\ln n / T)}$ is chosen, then $R_T \le 2\sqrt{(T \ln n)}$; hence $R_T/T \to 0$ and Hedge is no-regret.

The $\ln n$ in the bound is the natural logarithm of n; for $n = 3$, $\ln 3 \approx 1.1$. The logarithm grows very slowly: even with a thousand actions, $\ln 1000 \approx 6.9$. This shows that Hedge can learn quickly even among a very large number of actions.

The two terms of the bound measure two separate effects of η. The first term, $(\ln n)/\eta$, is the price of **slow learning**: if η is small, the algorithm notices the best action late. The second term, ηT, is the price of **overreaction**: if η is large, the algorithm reacts too strongly to every fluctuation and becomes exploitable like follow the leader. The best η is the value that minimizes the sum of the two terms. The derivative of the sum $f(\eta) = (\ln n)/\eta + \eta T$ is $f'(\eta) = -(\ln n)/\eta^2 + T$. Setting the derivative to zero gives $\eta^2 = (\ln n)/T$, that is, $\eta = \sqrt{(\ln n / T)}$. When this value is substituted, both terms equal $\sqrt{(T \ln n)}$:

$$(\ln n)/\sqrt{(\ln n / T)} = \sqrt{(T \ln n)} \text{ and } \sqrt{(\ln n / T)} \cdot T = \sqrt{(T \ln n)},$$

and their sum is $2\sqrt{(T \ln n)}$.

Example 13.7

Let us compare two games played with two actions ($n = 2$, $\ln 2 \approx 0.693$).

1. $T = 100$: $\eta = \sqrt{(0.693/100)} \approx 0.083$. The bound is $2\sqrt{(100 \cdot 0.693)} = 2\sqrt{69.3} \approx 16.7$. The regret per round is at most $16.7/100 \approx 0.17$.
2. $T = 10{,}000$: $\eta \approx 0.0083$. The bound is $2\sqrt{6931} \approx 166.5$. The regret per round is at most 0.017.

When the number of rounds grew 100-fold, the total bound grew only 10-fold ($\sqrt{100} = 10$); the per-round bound fell to one tenth. This is what no-regret means.

The concrete meaning of the bound becomes visible when the sequence that exploits follow the leader is fed to Hedge. Let the six-round sequence of Table 13.6 be given to Hedge with $\eta = 0.5$. In round 1 the probabilities are equal and the expected payoff is $0.5 \cdot 0.5 = 0.25$. In round 2 the weight of A is $e^{0.25} \approx 1.28$ and that of B is 1; $p(A) = 1.28/2.28 \approx 0.56$ and the expected payoff is $0.44 \cdot 1 \approx 0.44$. In the following rounds the probabilities oscillate between 0.56 and 0.44, and about 0.44 is earned each round. After six rounds Hedge's expected total is about 2.44 and its regret is $3 - 2.44 \approx 0.56$. Follow the leader had accumulated a regret of 2.5 on the same sequence. Because Hedge does not fully trust the small difference, the adversary's trick does not work.

Three observations will come up again and again in the rest of the chapter:

- **Randomization is essential.** Follow the leader is deterministic, so the opponent exploits it; Hedge plays a mixed strategy.
- **η is a tuning knob.** A large η reacts quickly but is sensitive to noise; a small η is slow but stable.

- **Equilibrium in games.** If everyone plays no-regret, the empirical distribution of the profiles played approaches the set of coarse correlated equilibria (CCE); if internal regret is minimized, it approaches the set of correlated equilibria (CE); and in zero-sum games the average strategies approach the minimax strategies.

13.3.4 Code trace: hedge()

The following function runs Hedge given the list `G`, which contains the payoff vector of each round, and the learning rate `eta`, and returns the algorithm's total expected payoff.

```
import math
def hedge(G, eta):
    w = [1.0] * len(G[0])
    total = 0.0
    for g in G:                          # each round
        W = sum(w)
        p = [x / W for x in w]
        total += sum(pi * gi for pi, gi in zip(p, g))
        w = [x*math.exp(eta*gi) for x, gi in zip(w, g)]
    return total
```

- `w` is the weight list; it starts with 1.0 for each action.
- In each round the sum of the weights `W` is computed first, and then each weight is divided by `W` to obtain the probabilities `p`.
- `total` is the sum of the expected payoffs. The expected payoff of this round is $\Sigma\, p_i \cdot g_i$; `zip(p, g)` pairs the two lists element by element.
- The last line is the update: each weight is multiplied by $e^{\eta \cdot g_i}$.

Let there be three experts (actions) A, B, C and let $\eta = 0.5$. In the first two rounds the payoffs are $g = [1, 0, 0.5]$, and in the next three rounds $g = [0, 1, 0.5]$. So A is good in the first two rounds, B is good in the next three, and C is always in the middle. The factors used are: for payoff 1, $e^{0.5} \approx 1.65$; for payoff 0.5, $e^{0.25} \approx 1.28$; for payoff 0, 1.

Round	w (A, B, C)	W	p	g	Σ p·g	total	new w
1	1; 1; 1	3.00	0.33; 0.33; 0.33	1; 0; 0.5	0.50	0.50	1.65; 1.00; 1.28
2	1.65; 1.00; 1.28	3.93	0.42; 0.25; 0.33	1; 0; 0.5	0.58	1.08	2.72; 1.00; 1.65
3	2.72; 1.00; 1.65	5.37	0.51; 0.19; 0.31	0; 1; 0.5	0.34	1.42	2.72; 1.65; 2.12
4	2.72; 1.65; 2.12	6.48	0.42; 0.25; 0.33	0; 1; 0.5	0.42	1.84	2.72; 2.72; 2.72
5	2.72; 2.72; 2.72	8.15	0.33; 0.33; 0.33	0; 1; 0.5	0.50	2.34	2.72; 4.48; 3.49

Table 13.7: Trace of `hedge(G, 0.5)`. *w* is the weights at the start of the round; "new w" is the update at the end of the round.

Example 13.8

1. **Round 1:** $W = 3$, probabilities equal (0.33). Expected payoff $⅓ \cdot 1 + ⅓ \cdot 0 + ⅓ \cdot 0.5 = 0.50$. Update: A $1 \cdot 1.65$, B unchanged, C $1 \cdot 1.28$.
2. **Round 2:** $W = 1.65 + 1 + 1.28 = 3.93$. $p(A) = 1.65/3.93 \approx 0.42$: A's share has grown. Expected payoff $0.42 \cdot 1 + 0.33 \cdot 0.5 \approx 0.58$; `total` becomes 1.08. A's weight rises to $1.65 \cdot 1.65 \approx 2.72 = e^1$.
3. **Round 3:** $W = 5.37$, $p(A) \approx 0.51$. But in this round B wins: expected payoff $0.19 \cdot 1 + 0.31 \cdot 0.5 \approx 0.34$. Because the algorithm adapts to the past, it is one round late.
4. **Round 4:** The weights of B and C have grown; $p \approx (0.42, 0.25, 0.33)$. Expected payoff 0.42. At the end of the round all three weights are equal to $e \approx 2.72$, because each action's total payoff is 2.
5. **Round 5:** The probabilities are equal again; expected payoff 0.50, `total` 2.34. The final weights are $(2.72, 4.48, 3.49) = (e^1, e^{1.5}, e^{1.25})$.

The totals of the fixed actions are A: 2, B: 3, C: 2.5. The best fixed action is B and the regret is $3 - 2.34 = 0.66$.

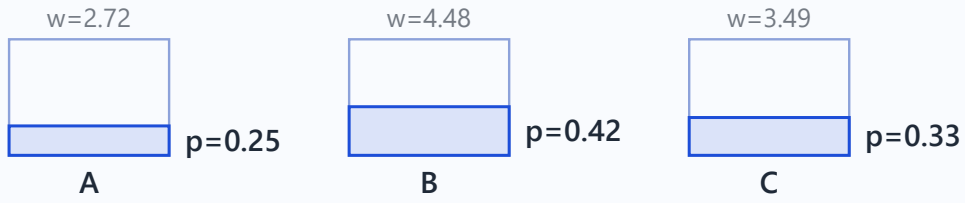


Figure 13.7: The end of the `hedge()` trace. After five rounds the weights are A: 2.72, B: 4.48, C: 3.49; the probabilities of the next round would be 0.25, 0.42 and 0.33.

Figure 13.7 shows how Hedge "remembers" through its weights. The final weights depend only on the total payoffs: $w_i = e^{0.5 \cdot G_i}$. The ordering is also the same as the ordering of the totals: B (3) > C (2.5) > A (2). The order in which the payoffs arrived does not change the weights; it changes only what the algorithm earned along the way.

Common mistake

- **Mistaking a weight for a probability:** a weight such as 2.72 is not a probability; the probability is found by dividing the weight by the total W.
- **Adding the payoff to the weight:** in Hedge the update is multiplicative ($w \cdot e^{\eta g}$), not additive ($w + \eta g$).

- **Mistaking the expected payoff for the realized payoff:** `total` is the sum of expected payoffs. In a real game one action is drawn in each round and the realized payoff fluctuates around this value.

13.3.5 When the opponent learns too

So far the payoff sequence was given from outside. In games, however, the opponent's behavior determines the payoffs, and the opponent can learn too. To see this difference, let us compare two scenarios in Rock-Paper-Scissors. The winner gets 1, a tie gets ½, and the loser gets 0; thus the payoffs lie in *[0, 1]*. Player 1 learns with Hedge and $\eta = 0.1$. (With a fixed η the bound in the theorem does not go to zero; the results below are the values observed for this duration and starting point.) The learning player updates in each round with the *expected* payoff of its actions.

Scenario 1: a fixed opponent. The opponent plays Rock with probability 0.5, Paper with 0.3 and Scissors with 0.2 in every round. The expected payoff of each of Player 1's actions can be computed.

Example 13.9

1. **Rock:** ties against Rock (½), loses to Paper (0), wins against Scissors (1). Expected payoff $0.5 \cdot ½ + 0.3 \cdot 0 + 0.2 \cdot 1 = 0.25 + 0.20 = 0.45$.
2. **Paper:** beats Rock, ties with Paper, loses to Scissors: $0.5 \cdot 1 + 0.3 \cdot ½ + 0 = 0.65$.
3. **Scissors:** loses to Rock, beats Paper, ties with Scissors: $0 + 0.3 \cdot 1 + 0.2 \cdot ½ = 0.40$.
4. The best response is Paper. In each round the weight of Paper grows by a factor $e^{0.1 \cdot 0.20}$ more than that of Rock and $e^{0.1 \cdot 0.25}$ more than that of Scissors. In Hedge started with equal weights, after n updates $p(\text{Paper}) = 1 / (1 + e^{-0.02\,n} + e^{-0.025\,n})$.
5. At round 100 this probability is about 0.82, and at round 300 it is 0.997.

Against a fixed opponent, Hedge converges to the best response. This is a natural consequence of no-regret: if the opponent does not change, the best fixed action is the best response, and the only way to keep regret small is to play it more and more often.

Scenario 2: a learning opponent. Now the opponent also uses Hedge, and the two players update at the same time. Player 1 starts with probabilities (0.5, 0.3, 0.2) and Player 2 with equal probabilities. As Player 1 drifts toward Paper, Scissors becomes more valuable for Player 2; as Player 2 drifts toward Scissors, Rock becomes more valuable for Player 1. The cycle goes on like this and the target keeps moving.

round	100	200	800	1000
P(Rock) in that round	0.19	0.46	0.14	0.19

round	100	200	800	1000
average P(Rock)	0.35	0.31	0.33	0.34

Table 13.8: Two Hedge players ($\eta = 0.1$), Rock-Paper-Scissors. Player 1's probability of Rock: the value in that round and the average up to that round.

Table 13.8 shows two different behaviors. The probability in that round (the last iterate) keeps rotating: it rises from 0.19 to 0.46 and falls to 0.14. Even after a thousand rounds it does not settle. The time average, on the other hand, approaches ⅓, the Rock component of (⅓, ⅓, ⅓), the game's only Nash equilibrium. Against a learning opponent, "reaching the target" is meaningless, because the target moves together with the opponent's behavior. What is stable is the average.

13.3.6 Hedge against itself

The same phenomenon is seen more clearly in a game with two actions. In Matching Pennies, let both players use Hedge ($\eta = 0.1$). The Matcher's initial weights are (H, T) = (0.8, 0.2), that is, $P(H) = 0.8$; the Mismatcher's are (0.5, 0.5). The winner gets 1 and the loser 0, and in every round both players update simultaneously with the expected payoffs of their actions.

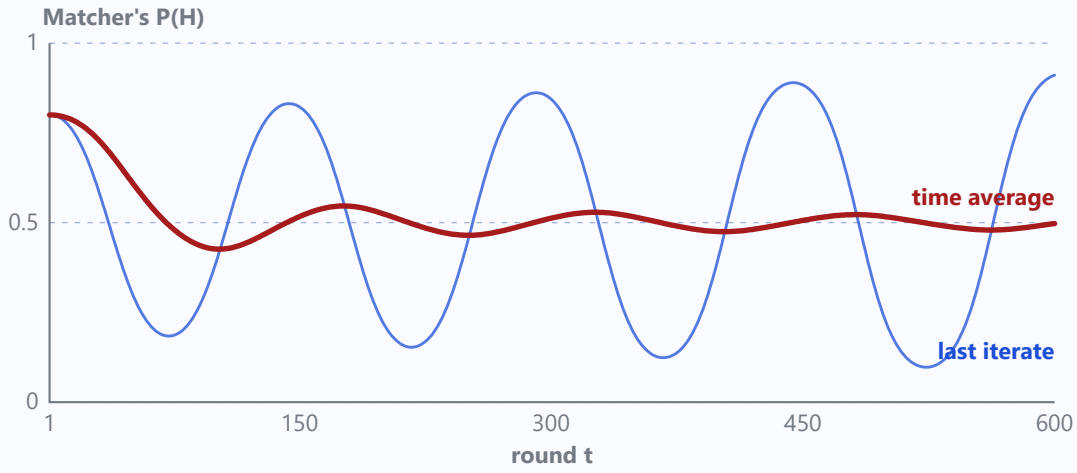


Figure 13.8: Two Hedge players in Matching Pennies. The thin blue curve is the Matcher's $P(H)$ in that round (last iterate), and the thick red curve is the time average. The last iterate makes large oscillations; the average settles at 0.5.

In Figure 13.8 the blue curve does not settle at the equilibrium; it makes oscillations that grow steadily between 0.1 and 0.9. The mechanism is similar to that in fictitious play. When the Matcher puts weight on H, T becomes more valuable for the Mismatcher; when the Mismatcher drifts to T, T becomes more valuable for the Matcher, and the cycle goes on. The red curve, which is the average of the $P(H)$ values since the start, approaches ½, the minimax strategy: it is 0.498 at round 600.

This observation has a theoretical counterpart. In a two-player zero-sum game, if both players use a no-regret algorithm, their average strategies approach the minimax strategies and the average payoff approaches the value of the game. The learning-based proof of von Neumann's minimax theorem, that is, the equality "max-min = min-max" in zero-sum games, rests on this observation. The idea of the proof is this: if the regret of both players goes to zero, then on average each earns as much as the best fixed action against the opponent's average strategy; when these two statements are combined, the gap between max-min and min-max is squeezed to zero.

Common mistake

Reading the sentence "no-regret algorithms converge to equilibrium" as a statement about the last iterate. The theory gives convergence only for the *time average*. In Figure 13.8 the last iterate is far from the equilibrium value even after 600 rounds.

13.3.7 From regret to equilibrium

Outside zero-sum games, where does the time average of no-regret learning go? The answer is two equilibrium concepts that are broader than Nash equilibrium. The easiest way to explain these concepts is to imagine a **recommendation device**. In each round the device draws an action profile from a probability distribution and tells each player only its own action. A traffic light at an intersection is such a device: it tells one driver "go" and the other "stop."

Definition 13.6

A distribution σ over action profiles is

- a **correlated equilibrium** (CE) if, for every player, following the recommendation is no worse than deviating, *after seeing its recommendation*;
- a **coarse correlated equilibrium** (CCE) if, for every player, committing *before seeing the recommendation* to "I will always play a', whatever the recommendation is" does not increase its expected payoff.

The difference between the two definitions lies in the moment of deviation. In a CE the player decides after seeing the recommendation; so deviations that depend on the recommendation, such as "play Straight when the recommendation is Swerve," are also taken into account. In a CCE the player, before seeing the recommendation, considers leaving the device altogether and switching to a fixed action. Since the conditions of a CE hold separately for each recommendation, they hold on average as well; so every CE is a CCE. The converse is not true: CCE is the weaker, that is, the broader concept.

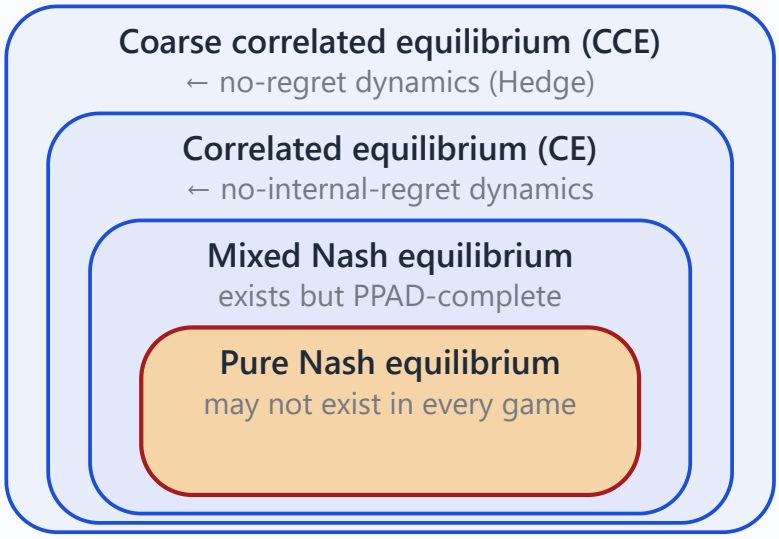


Figure 13.9: Nested sets of equilibrium concepts. A pure Nash equilibrium may not exist in every game; a mixed Nash equilibrium exists in every finite game but finding it is PPAD-complete; correlated equilibrium (CE) is found by internal-regret-free dynamics, and coarse correlated equilibrium (CCE) by no-regret dynamics such as Hedge.

The nested structure in Figure 13.9 is read as follows. Every pure Nash equilibrium is a mixed Nash equilibrium (one with probabilities 0 or 1). Every mixed Nash equilibrium is a CE: when the players throw independent dice, the player who sees its recommendation learns nothing new about the others and is already best-responding in equilibrium. Every CE is also a CCE. Going outward, the concepts get broader but easier to compute: CE and CCE are found quickly by linear programming, whereas a mixed Nash equilibrium is PPAD-complete.

Theorem 13.3 (No-regret learning and CCE)

If the external regret of all players repeatedly playing a game satisfies $R_T/T \le \varepsilon$, then the **time-average distribution** of the profiles played over T rounds (the fraction of rounds in which each profile was played) is an ε-CCE. For algorithms that minimize a stronger kind of regret, **internal regret**, the time average of the play approaches the set of CE, that is, it becomes an ε-CE as T grows (Foster and Vohra, 1997; Hart and Mas-Colell, 2000; Blum and Mansour, 2007).

Proof. Let the profiles played be $s^1, \ldots, s^T$ and let σ be their time-average distribution: drawing a profile from σ means choosing a random round among $1, \ldots, T$ and taking that round's profile. Player i's payoff in round t is $g_t(a) = u_i(a, s_{-i}^t)$. The regret condition says that for every fixed a'

$$\Sigma_t\, u_i(a', s_{-i}^t) - \Sigma_t\, u_i(s^t) \le R_T \le \varepsilon T$$

Dividing both sides by T turns the sums into averages. The average is the expected value under σ: $E_{s\sim\sigma}[u_i(a', s_{-i})] - E_{s\sim\sigma}[u_i(s)] \le \varepsilon$. This inequality says that the deviation "always play a' without seeing the recommendation" gains at most ε; it is the CCE definition satisfied up to a margin of ε. □

Internal (swap) regret asks, separately for each pair (a, a'), "what would I have earned if, in every round in which I played a, I had played a' instead?" This is the learning-theoretic counterpart of a deviation that depends on the recommendation: "play a' when the device says a." External regret compares all rounds with a single fixed action, so it is a coarser measure and leads to a coarser equilibrium.

Example 13.10

Consider the game of Chicken between two drivers at an intersection. The actions are Swerve (stop) and Straight (go); the payoffs are (Swerve, Swerve) = (0, 0), (Swerve, Straight) = (−1, 1), (Straight, Swerve) = (1, −1), (Straight, Straight) = (−10, −10). The traffic light recommends (Swerve, Straight) or (Straight, Swerve) with probability one half each.

1. **The driver who receives the recommendation "Swerve"** knows that the other has been told "Straight" for certain. Obeying gives −1; deviating to Straight gives −10. Obeying is better.
2. **The driver who receives the recommendation "Straight"** knows that the other has been told "Swerve" for certain. Obeying gives 1; deviating to Swerve gives 0. Obeying is better.
3. Obeying is best in both recommendations: the light is a correlated equilibrium. Each driver's expected payoff is $\tfrac{1}{2}\cdot(-1) + \tfrac{1}{2}\cdot 1 = 0$.
4. The CCE condition holds automatically as well. For example, the deviation "always Straight" gives $\tfrac{1}{2}\cdot 1 + \tfrac{1}{2}\cdot(-10) = -4.5$, since the other is told Swerve and Straight with probability one half each; this is worse than 0. The deviation "always Swerve" gives $\tfrac{1}{2}\cdot 0 + \tfrac{1}{2}\cdot(-1) = -0.5$, which is also worse than 0.

Disobeying the light gains nobody anything. No-regret learning finds equilibria of this kind without a central computer, simply by the players minimizing their own regrets.

Common mistake

- **Confusing CE with CCE:** Hedge minimizes only external regret and leads to CCE; reaching CE requires algorithms that minimize internal regret.
- **Mistaking the average distribution for each player's mixed strategy:** a CCE is a *joint* distribution over profiles; the players' actions may be linked (correlated). This distribution cannot always be written as a product of two independent mixed strategies.

13.4 The multi-armed bandit

Hedge sees the payoffs of *all* actions after every round. In the opening slot machine, however, only the result of the pulled arm is seen. This small difference has a big consequence: information is no longer free. The only way to gather information about an arm is to pull it; if that arm is bad, the pull is a loss of reward. The multi-armed bandit problem is the simplest model of this trade-off between information and reward.

13.4.1 The problem, regret, and the exploration–exploitation dilemma

Definition 13.7

Multi-armed bandit: in a slot machine with K arms, arm i gives a random reward from a distribution with mean μ_i that is unknown to the player. In each round one arm is pulled and only its reward is seen (partial feedback). With the best arm's mean $\mu^* = max_i\ \mu_i$, the regret over T rounds is measured by

$$R_T = T \cdot \mu^* - \Sigma_{t=1}^{T}\ \mu_{a_t}$$

where a_t is the arm pulled in round t.

This definition of regret carries the same idea as external regret in Hedge: comparison with the best fixed option (here, the best arm). The difference is that means are used rather than realized rewards. This filters out luck: pulling a bad arm and getting 1 by chance does not reduce the regret, because the calculation uses that arm's mean. Each round adds regret equal to how far the pulled arm falls behind the best arm, that is, $\mu^* - \mu_{a_t}$.

All the difficulty of the problem is concentrated in the **exploration–exploitation dilemma**. Exploration gathers information about new or little-tried arms. Exploitation uses what looks best so far. An algorithm that only exploits may get stuck on the wrong arm after an unlucky start and never escape. An algorithm that only explores pulls all arms equally and never uses what it has learned; it wastes reward.

Bandit algorithms keep two numbers for each arm: the number of times the arm has been pulled, N_i, and the average of the observed rewards, Q_i. The average is not recomputed from scratch each time a new reward arrives; it is updated with the **incremental mean** formula:

$$Q \leftarrow Q + (r - Q)/N$$

Here N is the number of pulls after the new reward has been counted. The correctness of the formula is seen with a short calculation. If the average of the previous $N - 1$ rewards is Q, their sum is $(N - 1)Q$. When the new reward is added, the new average is $((N - 1)Q + r)/N = (NQ - Q + r)/N = Q + (r - Q)/N$. The interpretation of the formula is also clear: the estimate moves by $1/N$ in the direction of the "surprise" $r - Q$. The estimate of an arm that has been pulled many times changes little with a single observation.

Example 13.11

Let an arm's rewards be, in order, 1, 1, 1, 0, 1.

1. $N = 1$: $Q = 0 + (1 - 0)/1 = 1$.
2. $N = 2$: $Q = 1 + (1 - 1)/2 = 1$. $N = 3$: again 1.
3. $N = 4$: $Q = 1 + (0 - 1)/4 = 0.75$.
4. $N = 5$: $Q = 0.75 + (1 - 0.75)/5 = 0.75 + 0.05 = 0.80$.

The result is the same as the direct average of the five rewards ($4/5 = 0.80$).

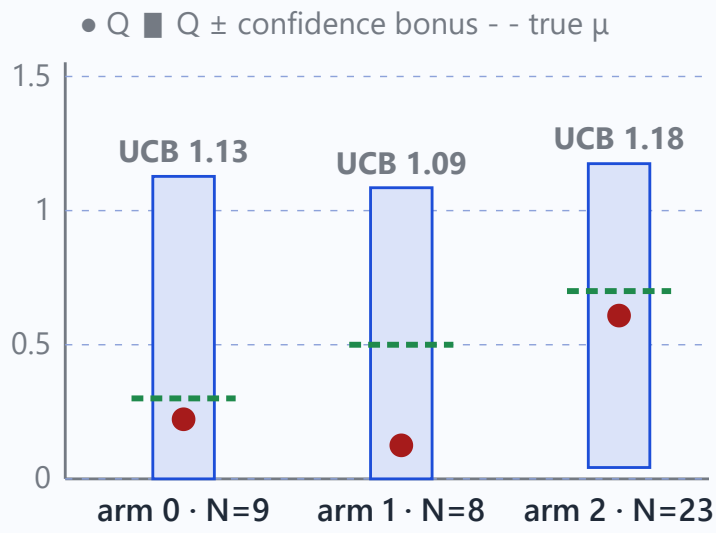


Figure 13.10: The UCB1 algorithm, true means $\mu = (0.3, 0.5, 0.7)$, after 40 rounds. Red dots are the estimates Q, blue bars are $Q \pm$ *confidence bonus*, green dashed lines are the true μ. Arms pulled few times (arm 0: 9, arm 1: 8 pulls) have wide intervals; arm 2, pulled often (23 pulls), has a narrow one.

Figure 13.10 shows the basic idea of the UCB1 algorithm (13.4.3). The estimate of a rarely tried arm is uncertain, so its confidence interval is wide. The upper end of the interval, that is, "the best value this arm could have," makes the arm worth trying again. Even though its estimate is low, arm 1's upper end is 1.09, close to arm 2's 1.18.

The application area of the bandit problem extends far beyond game theory: recommender systems that decide which ad or news item to show, A/B tests that compare two designs on websites, clinical trials in which the allocation of patients to the better treatment is adjusted during the trial, and price experiments in which a product is tried at different prices.

13.4.2 Code trace: ε-greedy

The simplest bandit algorithm is **ε-greedy**. In each round, with a small probability ε it pulls a random arm (exploration); with the remaining probability $1 - \varepsilon$ it pulls the arm with the largest Q (exploitation). The name "greedy" comes from the exploitation step blindly trusting the current best estimate.

```
def egreedy(mu, T, eps, rng):
    N, Q = [0] * len(mu), [0.0] * len(mu)
    for t in range(1, T + 1):
        if rng.random() < eps:           # explore
            a = rng.randrange(len(mu))
        else:                            # exploit
            a = Q.index(max(Q))
        r = 1 if rng.random() < mu[a] else 0
        N[a] += 1
        Q[a] += (r - Q[a]) / N[a]
    return N, Q
```

- `mu` holds the true means of the arms. The algorithm uses this list only to generate rewards; it does not look at `mu` when deciding.
- `rng.random()` produces a random number between 0 and 1. If the number is smaller than `eps`, the algorithm explores and picks a random arm with `rng.randrange(len(mu))`.
- When exploiting, `Q.index(max(Q))` finds the arm with the largest estimate. In case of a tie, the *first* arm in the list is chosen.
- The reward is a coin flip determined by the arm's mean: if a new random number is smaller than `mu[a]` the reward is 1, otherwise 0.
- The last two lines are the counter and the incremental mean update.

Let the call be `egreedy([0.3, 0.5, 0.7], 8, 0.2, random.Random(11))`. In the code the arms are numbered from 0: arms 0, 1, 2 correspond to Arms 1, 2, 3 in the opening game. Since the random number generator is initialized with seed 11, the call gives the same result every time it is run; the trace below is therefore reproducible.

t	Random number	Decision	Arm	r	N	Q
1	0.452	exploit (first on tie)	0	0	[1, 0, 0]	[0.00, 0.00, 0.00]
2	0.924	exploit (first on tie)	0	0	[2, 0, 0]	[0.00, 0.00, 0.00]
3	0.508	exploit (first on tie)	0	0	[3, 0, 0]	[0.00, 0.00, 0.00]

t	Random number	Decision	Arm	r	N	Q
4	0.185	explore	2	1	[3, 0, 1]	[0.00, 0.00, 1.00]
5	0.614	exploit	2	1	[3, 0, 2]	[0.00, 0.00, 1.00]
6	0.447	exploit	2	1	[3, 0, 3]	[0.00, 0.00, 1.00]
7	0.539	exploit	2	0	[3, 0, 4]	[0.00, 0.00, 0.75]
8	0.634	exploit	2	1	[3, 0, 5]	[0.00, 0.00, 0.80]

Table 13.9: Trace of `egreedy` ($\varepsilon = 0.2$, seed 11). "Random number" is the first number used in the exploration decision.

Example 13.12

1. **t = 1–3:** The random numbers are larger than 0.2; all three are exploitation. Since all estimates are 0, the first arm on the tie, arm 0, is chosen. Arm 0 gives 0 all three times; its estimate stays at 0 and the tie is never broken.
2. **t = 4:** The random number 0.185 < 0.2: exploration. The randomly chosen arm is 2 and it gives 1. *N[2] = 1*, *Q[2] = 1.00*.
3. **t = 5–6:** Exploitation; the largest estimate is now at arm 2. Both pulls give 1; *Q[2]* stays at 1.
4. **t = 7:** Arm 2 gives 0 this time: *Q[2] = 1 + (0 − 1)/4 = 0.75*.
5. **t = 8:** Arm 2 gives 1: *Q[2] = 0.75 + (1 − 0.75)/5 = 0.80*.

The function returns *N = [3, 0, 5]*. The regret over the eight rounds is $8 \cdot 0.7 - (3 \cdot 0.3 + 5 \cdot 0.7) = 5.6 - 4.4 = 1.2$; all of it comes from the first three rounds spent on arm 0.



Figure 13.11: The end of the ε-greedy trace. Arm 0 has been pulled three times, its estimate is 0; arm 1 has never been pulled; arm 2 has been pulled five times, its estimate is 0.80. The green dashed lines are the true means (0.3, 0.5, 0.7).

The trace shows both faces of ε-greedy at once. The negative face: for the first three rounds it was stuck on arm 0, because all pulls gave 0 and the tie rule chose the first arm. Without exploration the algorithm could have stayed on arm 0 forever; even if arm 0 gave 1 once, its estimate would exceed the 0 of the other arms and exploitation would keep choosing it. The positive face: a single exploration found the best arm. Figure 13.11 also shows a shortcoming: arm 1 was never tried. In eight rounds this does not matter, but the exploration of ε-greedy is blind randomness; it does not specifically look for little-known arms.

13.4.3 Code trace: UCB1

The exploration of ε-greedy is "blind": in an exploration round every arm is chosen with equal probability, both the arm already known to be best and an arm never tried. **UCB1** (Upper Confidence Bound) directs exploration. Each arm has a score:

$$UCB_i = Q_i + \sqrt{(2 \ln t / N_i)}$$

Here t is the current round number and N_i is the number of times the arm has been pulled. In each round the arm with the largest score is pulled. The score of an arm that has never been pulled is taken to be infinite; so in the first rounds every arm is tried once.

The score has two terms. The first term, Q_i, is exploitation: an arm that looks good gets a high score. The second term, $\sqrt{(2 \ln t / N_i)}$, is the **confidence bonus** and represents exploration. The bonus shrinks as N_i grows: the estimate of a much-pulled arm can be trusted. The bonus grows very slowly (logarithmically) as t grows: the bonus of an arm that has not been pulled for a long time slowly increases, and eventually the arm is tried again. This idea is called **optimism in the face of uncertainty**: each arm is evaluated at the most optimistic value that does not contradict the data. If a bad arm is chosen on an optimistic evaluation, it soon shows its true face and its bonus shrinks; a good arm, on the other hand, deserves the optimistic evaluation.

```
import math
def ucb(mu, T, rng):
    K = len(mu)
    N, Q = [0] * K, [0.0] * K
    for t in range(1, T + 1):
        b = [math.inf if n == 0 else
             math.sqrt(2*math.log(t)/n) for n in N]
        a = max(range(K), key=lambda i: Q[i] + b[i])
        r = 1 if rng.random() < mu[a] else 0
        N[a] += 1
        Q[a] += (r - Q[a]) / N[a]
    return N, Q
```

- `b` holds each arm's confidence bonus. `math.inf` is the infinite bonus of an arm that has never been pulled.
- `max(range(K), key=lambda i: Q[i] + b[i])` chooses the arm with the largest score $Q + b$; on a tie the arm with the smaller index wins.
- The reward and the update are as in ε-greedy.

Let the call be `ucb([0.3, 0.5, 0.7], 8, random.Random(3))`. Table 13.10 gives the bonuses, scores and choice in each round.

t	bonus b	UCB = Q + b	Arm	r	Q (after)	N (after)
1	∞, ∞, ∞	∞, ∞, ∞	0	1	1.00; 0; 0	1, 0, 0
2	1.18; ∞; ∞	2.18; ∞; ∞	1	0	1.00; 0.00; 0	1, 1, 0
3	1.48; 1.48; ∞	2.48; 1.48; ∞	2	1	1.00; 0.00; 1.00	1, 1, 1
4	1.67; 1.67; 1.67	2.67; 1.67; 2.67	0	0	0.50; 0.00; 1.00	2, 1, 1
5	1.27; 1.79; 1.79	1.77; 1.79; 2.79	2	1	0.50; 0.00; 1.00	2, 1, 2
6	1.34; 1.89; 1.34	1.84; 1.89; 2.34	2	1	0.50; 0.00; 1.00	2, 1, 3
7	1.39; 1.97; 1.14	1.89; 1.97; 2.14	2	1	0.50; 0.00; 1.00	2, 1, 4
8	1.44; 2.04; 1.02	1.94; 2.04; 2.02	1	0	0.50; 0.00; 1.00	2, 2, 4

Table 13.10: Trace of `ucb` (seed 3). The bonuses and scores are the values before the choice; Q and N are the values after the choice.

Example 13.13

1. **t = 1, 2, 3:** The bonus of arms never tried is infinite; the arms are pulled once each in order. The rewards are 1, 0, 1.
2. **t = 4:** Every arm has been pulled once: the bonus is $\sqrt{(2 \ln 4 / 1)} = \sqrt{2.77} \approx 1.67$, the same for all three arms. The scores are $1 + 1.67 = 2.67$, $0 + 1.67 = 1.67$, 2.67. Arms 0 and 2 tie; the smaller index, arm 0, is chosen and gives 0. $Q[0] = 0.50$.
3. **t = 5:** Arm 0 has been pulled twice: its bonus is $\sqrt{(2 \ln 5 / 2)} = \sqrt{1.61} \approx 1.27$. The others' bonus is $\sqrt{(2 \ln 5)} \approx 1.79$. The scores are 1.77, 1.79, 2.79. Arm 2 is chosen and gives 1.
4. **t = 6, 7:** Arm 2 keeps winning. But its bonus shrinks with each pull (1.34 → 1.14), while arm 1's bonus grows (1.89 → 1.97).
5. **t = 8:** Arm 1's bonus has reached $\sqrt{(2 \ln 8 / 1)} = \sqrt{4.16} \approx 2.04$; its score is 2.04. Arm 2's score is $1 + \sqrt{(4.16/4)} = 1 + 1.02 = 2.02$. Arm 1 edges ahead and is tried again.

The function returns $N = [2, 2, 4]$. Arm 1's single trial gave 0 and its estimate is 0; it is nevertheless tried again in round 8. Exploration arises from the uncertainty itself, without the need for a separate parameter.



Figure 13.12: The end of the UCB1 trace: arm 0 was pulled twice (Q = 0.50), arm 1 twice (Q = 0), arm 2 four times (Q = 1.00).

Comparing the two traces is instructive. ε-greedy never tried arm 1; UCB1 tried every arm and revisited even the arm with a low estimate once it had not been pulled for a long time. Because UCB1 explored more in eight rounds, it pulled non-best arms more often (four rounds against three). But since two of those rounds went to the middle arm (arm 1), its regret is again $8 \cdot 0.7 - (2 \cdot 0.3 + 2 \cdot 0.5 + 4 \cdot 0.7) = 5.6 - 4.4 = 1.2$. In the short run the two algorithms are in the same place; the difference appears in the long run.

Common mistake

- **Computing the bonus with the wrong t:** t is the current round number (starting from 1), not the number of times the arm has been pulled. At $t = 8$, $\ln 8$ is used in the bonus of all arms.
- **Choosing the largest Q:** UCB1 chooses not the largest estimate but the largest *upper bound*. In round 8 the largest estimate is at arm 2 (1.00) but the chosen arm is arm 1.
- **Using $\log_{10}$ instead of ln:** the $\ln$ in the formula is the natural logarithm; $\ln 8 \approx 2.08$, $\log_{10} 8 \approx 0.90$.

13.4.4 Comparison: 5000 rounds, 100 repetitions

The eight-round traces show how the algorithms work, but not which is better. Since the rewards are random, a single experiment is misleading. The comparison is done as follows: the arms are again $\mu = (0.3, 0.5, 0.7)$; each algorithm is run for 5000 rounds; the experiment is repeated with 100 different seeds from 0 to 99, and the cumulative regrets are averaged. In this comparison ε-greedy runs with $\varepsilon = 0.1$.

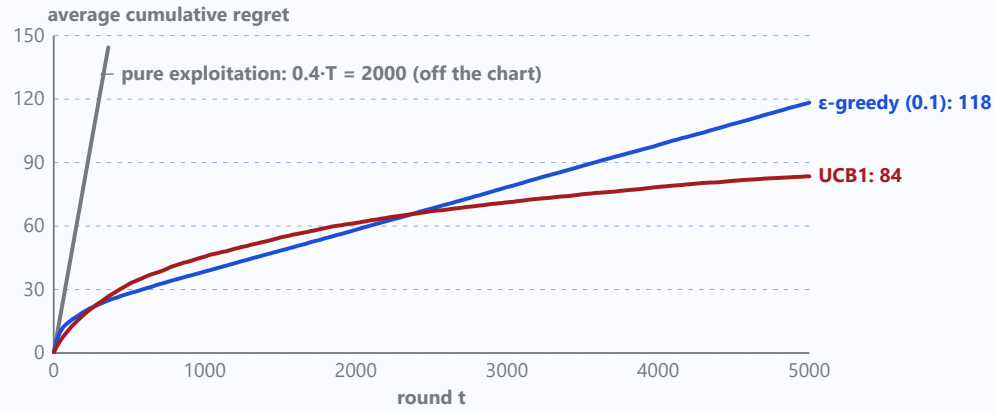


Figure 13.13: Cumulative regret averaged over 5000 rounds and 100 repetitions. Gray: pure exploitation ($0.4 \cdot T$, 2000 at 5000 rounds; it runs off the chart). Blue: ε-greedy ($\varepsilon = 0.1$), 118 at 5000 rounds. Red: UCB1, 84 at 5000 rounds.

Figure 13.13 shows three different growth rates.

- **Pure exploitation** ($\varepsilon = 0$): since all estimates start at 0, the first round pulls arm 0. If arm 0 gives 1 once, its estimate becomes positive and the other arms are never tried; if it always gives 0, the tie again chooses arm 0. In every case the algorithm gets stuck on arm 0. Each round adds $0.7 - 0.3 = 0.4$ regret: $R_T = 0.4 \cdot T$, linear.
- **ε-greedy**: even after finding the best arm, it explores in 10% of the rounds. In an exploration round the arms are chosen with equal probability and the average loss is $(0.4 + 0.2 + 0)/3 = 0.2$. So each round adds about $0.1 \cdot 0.2 = 0.02$ regret; over 5000 rounds this comes to about 100. Together with the initial learning losses the average is 118. The regret grows linearly, but with a small slope.
- **UCB1**: it explores more at the start. Its regret is higher than ε-greedy's between about rounds 270 and 2,370. After that it stays lower, because UCB1's regret is **logarithmic**, that is, it grows very slowly (Auer, Cesa-Bianchi and Fischer, 2002). Since the confidence bonus of a bad arm is $\sqrt{(2 \ln t / N)}$, for that arm to be tried again N only has to grow in proportion to $\ln t$; the number of pulls spent on bad arms therefore grows only like $\ln t$.

How slowly logarithmic growth proceeds can be seen from numbers: $\ln 1000 \approx 6.9$, $\ln 1{,}000{,}000 \approx 13.8$. When the number of rounds grows a thousandfold, the logarithm only doubles. With linear regret, a thousandfold increase in the number of rounds raises the regret a thousandfold. In long games this difference is decisive.

ε is a tuning parameter, and three different values are used in this chapter: 0.2 so that exploration is visible in the short trace, 0.1 in the long comparison, and 0.15 in the Multi-Armed Bandit game. A small ε means less exploration but slower correction after a wrong start. Logarithmic regret cannot be obtained with a fixed ε; for that, ε has to be decreased over time.

> **Common mistake**
>
> Seeing in the chart that ε-greedy has less regret at round 1000 and concluding that "ε-greedy is better." The answer to the comparison depends on the length of the game. In short games a simple ε-greedy can do well; in long games an algorithm with logarithmic regret is bound to overtake one with linear regret.

13.4.5 Back to the game: the price of exploration

We can return to the opening game with the theory's answer. For arms $\mu = (0.3, 0.5, 0.7)$, the expected reward and regret of different strategies over 100 rounds are given in Table 13.11.

Strategy	Reward	Regret
always random (exploration only)	50	20
sticking to the 0.5 arm	50	20
ε-greedy, ε = 0.15 (after finding the best arm)	67	3

Strategy	Reward	Regret
always the best arm	70	0

Table 13.11: Expected reward and regret over 100 rounds, $\mu = (0.3, 0.5, 0.7)$.

Example 13.14

1. **Always random:** in each round the arms are chosen with equal probability; the expected reward per round is $(0.3 + 0.5 + 0.7)/3 = 0.5$. Over 100 rounds, 50; regret $70 - 50 = 20$.
2. **Sticking to the 0.5 arm:** 0.5 per round; again 50 and 20.
3. **ε-greedy (ε = 0.15), after finding the best arm:** per round $0.85 \cdot 0.7 + 0.15 \cdot 0.5 = 0.595 + 0.075 = 0.67$. Over 100 rounds, 67; regret 3. Here 0.5 is the expected reward of the arm chosen at random in an exploration round.
4. **Always the best arm:** 70; this is the limit that someone who knows the true means can reach.

The table clearly shows that exploration has a price. Knowing the best arm is worth 3 points; ε-greedy loses these 3 points because it keeps exploring even after finding the best arm. UCB1 directs exploration toward little-tried arms, so it reduces this price over time; its regret grows logarithmically.

The opening question. In the situation of Figure 13.1, Arm 3's single failed pull gives very little information. The next round is $t = 9$ and $2 \ln 9 \approx 4.39$. The UCB1 scores are:

- Arm 1 ($Q = 0.67$, $N = 3$): $0.67 + \sqrt{(4.39/3)} = 0.67 + 1.21 \approx 1.88$.
- Arm 2 ($Q = 0.25$, $N = 4$): $0.25 + \sqrt{(4.39/4)} = 0.25 + 1.05 \approx 1.30$.
- Arm 3 ($Q = 0$, $N = 1$): $0 + \sqrt{4.39} \approx 2.10$.

UCB1 chooses Arm 3. A single failure is not enough evidence to set an arm aside.

How do humans decide? In a four-armed bandit experiment, the choices of human subjects were explained better by probabilistic choice according to value differences than by UCB-like exploration that gives a bonus to uncertainty (Daw et al., 2006). This choice rule is called **softmax choice**: the probability of choosing an arm is proportional to an exponential function of its estimated value. This rule has the same form as $p_i \propto e^{\eta \cdot G_i}$ in Hedge.

Common mistake

Ranking algorithms from a single experiment. The rewards are random; with a lucky seed ε-greedy can easily beat UCB1, and with an unlucky seed UCB1 can beat ε-greedy. Comparisons are always made by averaging many experiments.

13.5 Reinforcement learning

In the bandit problem each round is independent of the others: whichever arm is pulled today, tomorrow the machine is the same machine. In many real problems, however, today's action changes tomorrow's situation. In chess a move changes the board; a robot's step changes its position. This section treats learning problems in which actions also affect the future state.

13.5.1 From bandits to reinforcement learning: Q-learning

The bandit is a single-state problem. In **reinforcement learning** (RL), by contrast, an action determines the next **state** as well as the reward. The state is where the agent is: the arrangement on a board, a cell in a maze, or the joint angles of a robot. The value of an action can no longer be measured only by the reward it gets immediately; the rewards that the state it reaches will provide in the future must also be counted. A state one step away from the goal is valuable even if it gives no reward.

Q-learning (Watkins, 1989) keeps a value estimate for each state–action pair: $Q(s, a)$ answers the question "if action a is taken in state s and play continues optimally afterward, what is the total payoff?" When the agent takes action a in state s, receives reward r and moves to state s', it updates the estimate as follows:

Definition 13.8

$$Q(s, a) \leftarrow Q(s, a) + \alpha \left[r + \gamma \cdot \max_{a'} Q(s', a') - Q(s, a) \right]$$

α is the **learning rate**: it determines how much the new information is trusted. γ is the **discount factor**: it determines the present value of future reward ($0 \le \gamma < 1$). The first two terms inside the square brackets, $r + \gamma \cdot \max Q(s', \cdot)$, are the **target** of the update.

The structure of the formula is the same as the bandit's incremental mean: "new estimate = old estimate + step × (target − old estimate)." The difference is in the target. In the bandit the target was just the observed reward. In Q-learning the target is the discounted sum of the observed reward and the best value of the state reached. The term $\max_{a'} Q(s', a')$ carries the assumption that the best action will be taken after reaching s'. This means that Q-learning learns by using its own estimates: part of the target is itself an estimate.

The meaning of γ becomes clear with an example. If $\gamma = 0.9$, 1 point arriving one step later is worth 0.9 today, and 1 point arriving two steps later is worth $0.9^2 = 0.81$. If γ is near zero the agent looks only at the immediate reward (it is myopic); if it is near 1 it cares about the distant future as well.

Example 13.15

Let $Q(s, a) = 2$, $r = 1$, $\gamma = 0.9$, $\max Q(s', \cdot) = 4$, $\alpha = 0.5$.

1. Target: $r + \gamma \cdot \max Q(s', \cdot) = 1 + 0.9 \cdot 4 = 1 + 3.6 = 4.6$.
2. Difference (error): $4.6 - 2 = 2.6$. The estimate has fallen short of the target.
3. Update: $Q \leftarrow 2 + 0.5 \cdot 2.6 = 2 + 1.3 = 3.3$.

The estimate moved halfway ($\alpha = 0.5$) toward the target. With $\alpha = 1$ the estimate would jump straight to the target (4.6); with $\alpha = 0.1$ it would rise only to 2.26.

In a **multi-agent setting** an important weakness of Q-learning appears. The convergence proof of single-agent Q-learning assumes an environment in which transitions and rewards do not change over time, that is, a **stationary** environment. In a multi-agent setting, the outcomes an agent sees depend on the behavior of the other agents. As the other agents learn too, the rules of the game keep changing in the agent's eyes: an action that was good yesterday may be bad today. The environment is **non-stationary**, and the single-agent convergence guarantees no longer hold. One way to manage this problem is self-play (AlphaZero): the agent plays against copies of itself, so the opponent improves together with the agent.

13.5.2 Code trace: q_learn()

The environment is a corridor with four cells: s_0, s_1, s_2, s_3. In every cell there are two actions: left and right. s_3 is the goal; the transition that reaches it gives reward 1 and the episode (one attempt) ends. The reward of the other transitions is 0. The function processes pre-recorded episodes (each a list of (s, a, r, s') quadruples) in order.

```
def q_learn(episodes, alpha, gamma):
    Q = [[0.0, 0.0] for s in range(4)]  # [left, right]
    for ep in episodes:
        for s, a, r, s2 in ep:
            best = 0 if s2 == 3 else max(Q[s2])
            target = r + gamma * best
            Q[s][a] += alpha * (target - Q[s][a])
    return Q
```

- `Q` is a table with four rows; each row holds the values of the actions [left, right]. Action 0 is left, 1 is right.
- `best` is the best value of the state reached. If the goal has been reached (`s2 == 3`), the episode ends and there is no future; the value is taken as 0.
- `target` is $r + \gamma \cdot best$; the last line is the Q-learning update.

Let the call be `q_learn(ep, 0.5, 0.9)`. `ep` consists of three episodes. In the first two the agent starts at s_0 and goes right three times. In the third, at s_1 it goes left once, then proceeds right again.

Episode	Transition	r	best	target	Update
1	s0 → right → s1	0	0.000	0.000	Q(s0,right) = 0 + 0.5·(0 − 0) = 0.000
1	s1 → right → s2	0	0.000	0.000	Q(s1,right) = 0.000
1	s2 → right → s3	1	0 (goal)	1.000	Q(s2,right) = 0 + 0.5·(1 − 0) = 0.500
2	s0 → right → s1	0	0.000	0.000	Q(s0,right) = 0.000
2	s1 → right → s2	0	0.500	0.450	Q(s1,right) = 0 + 0.5·(0.450 − 0) = 0.225
2	s2 → right → s3	1	0 (goal)	1.000	Q(s2,right) = 0.5 + 0.5·(1 − 0.5) = 0.750
3	s0 → right → s1	0	0.225	0.203	Q(s0,right) = 0 + 0.5·(0.203 − 0) = 0.101
3	s1 → left → s0	0	0.101	0.091	Q(s1,left) = 0 + 0.5·(0.091 − 0) = 0.046
3	s0 → right → s1	0	0.225	0.203	Q(s0,right) = 0.101 + 0.5·(0.203 − 0.101) = 0.152
3	s1 → right → s2	0	0.750	0.675	Q(s1,right) = 0.225 + 0.5·(0.675 − 0.225) = 0.450
3	s2 → right → s3	1	0 (goal)	1.000	Q(s2,right) = 0.75 + 0.5·(1 − 0.75) = 0.875

Table 13.12: Trace of `q_learn` ($\alpha = 0.5$, $\gamma = 0.9$). Each row is one transition and one update.

Example 13.16

1. **Episode 1:** In the first two transitions the reward is 0 and the values of the states reached are 0; nothing is learned. Only the last transition brings reward 1, and *Q(s2,right) = 0.5*. For now, the reward information has reached only one step before the goal.
2. **Episode 2:** In the transition from *s1* to *s2*, *best = max Q(s2, ·) = 0.5*. The target is *0 + 0.9 · 0.5 = 0.45* and the update is *0.5 · 0.45 = 0.225*. The reward has propagated one step further back. *s0* is still 0, because its update was made before *Q(s1, ·)* increased.
3. **Episode 3, first transition:** *best = 0.225*, the target is *0.9 · 0.225 ≈ 0.203*, and *Q(s0,right) ≈ 0.101*. The reward information has reached the starting cell.
4. **The deviation to the left:** when the agent goes left from *s1*, *best = max Q(s0, ·) = 0.101*, the target is 0.091, and *Q(s1,left) ≈ 0.046*. Going left also gained a little value, because left leads to *s0*, whose value is no longer zero.
5. **The remaining transitions:** The agent proceeds right again and each update raises the values a little more: *Q(s0,right) ≈ 0.152*, *Q(s1,right) = 0.450*, *Q(s2,right) = 0.875*.



Figure 13.14: The Q table after three episodes. In each cell the values of the actions left (←) and right (→): *s0*: 0.000 / 0.152; *s1*: 0.046 / 0.450; *s2*: 0.000 / 0.875. *s3* is the goal (r = 1, the episode ends).

Figure 13.14 summarizes the basic behavior of Q-learning: reward propagates backward from the goal. $Q(s2,right) = 0.875 \rightarrow Q(s1,right) = 0.450 \rightarrow Q(s0,right) = 0.152$. Since $Q(right) > Q(left)$ in every cell, the **greedy policy**, which chooses the action with the largest Q in every state, says "always go right." The agent has learned this policy although it sees reward only at the last step.

Where would the values go if the same episodes were repeated forever? When the agent always goes right, the target of $Q(s2,right)$ is 1 every time, so $Q(s2,right) \rightarrow 1$. The target of $Q(s1,right)$ is $0.9 \cdot Q(s2,right)$, and it goes to 0.9. $Q(s0,right)$ goes to $0.9^2 = 0.81$. The values decrease with distance from the goal as powers of γ: distant reward is discounted.

Common mistake

- **Using max Q in the goal state:** the episode ends at the goal; the goal has no future, so $best = 0$.
- **Mixing up the order of updates:** within an episode the updates are made in the order of the transitions. When *s0* is updated, $Q(s1, \cdot)$ still has its old value from before that episode; this is why the reward propagates only one step back per episode.
- **Adding the target instead of the difference:** $Q \leftarrow Q + \alpha \cdot target$ is wrong; what is added is $\alpha \cdot (target - Q)$.

13.5.3 Beyond Q-learning: methods and milestones

The algorithms so far are the simplest members of a broad family. Some of the family's other important members are the following:

- **Thompson sampling** (Thompson, 1933): a Bayesian method for bandits. For each arm it keeps a probability distribution of the form "its mean could be this"; this distribution is a posterior updated with the observations. In each round a sample is drawn from each arm's distribution and the arm with the largest sample is played. An uncertain arm is explored automatically because it sometimes gives a high sample; it is a probabilistic counterpart of the optimism in UCB1.
- **Win-Stay Lose-Shift** (WSLS; Nowak and Sigmund, 1993): a rule for the Prisoner's Dilemma. If 3 or 5 points were received, the same move is repeated; if 0 or 1 were received, the move is changed. After a single mistake it returns to cooperation in two rounds.

- **EXP3**: Hedge that sees only the reward of the pulled arm; it is designed for the adversarial bandit. Even if a malicious opponent chooses the rewards, its regret grows more slowly than T (Auer, Cesa-Bianchi, Freund and Schapire, 2002).
- **SARSA and DQN**: relatives of Q-learning. DQN uses a neural network instead of a Q table; this lets it work on problems with too many states to fit in a table (for example, Atari games from screen images) (Mnih et al., 2015).
- **CFR** (counterfactual regret minimization): a regret-based method for poker. It keeps a separate regret at each decision point of the game and minimizes them (Zinkevich et al., 2007).

Year	System	Method
1992	TD-Gammon (backgammon)	self-play
2017–19	Libratus, Pluribus (poker)	CFR (regret)
2018	AlphaZero (chess, Go)	self-play

Table 13.13: Milestones of learning in games.

Games are a good laboratory for reinforcement learning: the rules are explicit, the reward is clear (winning or losing), and the game can be repeated millions of times. Table 13.13 shows that two approaches stand out. In games of perfect information (backgammon, chess, Go), self-play succeeded. In games of imperfect information, especially poker, regret-based methods came to the fore because the opponent's cards are not visible. The success in poker is directly connected to the theory of this chapter: because CFR minimizes regret in a two-player zero-sum game, its time average approaches the minimax strategy.

13.6 Evolutionary dynamics

In all the models so far there was a learning individual: a player that keeps counters, updates weights, makes estimates. In evolutionary game theory the individual learns nothing. Each individual is born with a single strategy and plays it for its whole life. What changes is the composition of the population: individuals carrying the successful strategy leave more offspring, and that strategy spreads. There is no rationality, there is selection. The surprising thing is that this "mindless" process often arrives where the Nash equilibrium predicts.

13.6.1 Replicator dynamics and the ESS

The model is as follows. In a large population, individuals are randomly paired and play a game against each other. Each individual plays a single strategy and passes it on to its offspring. The success of a strategy is measured by the average payoff of an individual playing it against an opponent drawn at random from the population.

Definition 13.9

Let x_i be the share of strategy i in the population. Let f_i be the average payoff of strategy i, called its **fitness**, and let $\bar{f} = \sum_j x_j f_j$ be the population's average payoff. The discrete **replicator dynamics** updates the shares in each generation as follows:

$$\text{new share} = \text{old share} \times f_i / \bar{f}, \text{ that is, } x_i' = x_i \cdot f_i / \bar{f}$$

The formula follows from the assumption that the number of offspring is proportional to fitness. Let n_i individuals in the population play strategy i, and let each leave f_i offspring. In the new generation $n_i f_i$ individuals play i. The new share is the ratio of this to the total number of offspring: $n_i f_i / \sum_j n_j f_j$. Dividing numerator and denominator by the total population n gives $x_i f_i / \sum_j x_j f_j = x_i f_i / \bar{f}$.

The formula's interpretation is a single sentence: **a strategy that earns above the average spreads**. If $f_i > \bar{f}$ then $f_i/\bar{f} > 1$ and the share grows; if $f_i < \bar{f}$ the share shrinks. The shares always remain a distribution that sums to 1. For the formula to make sense, the fitnesses must be positive: a negative fitness would mean a negative number of offspring. For this reason a constant **baseline** (3) is added to every payoff here. Adding a baseline does not change the ranking of the strategies or the rest points; it only affects the speed of the dynamics. (In the continuous-time version one writes $\dot{x}_i = x_i(f_i - \bar{f})$; the discrete form suffices here.)

The equilibrium concept of the evolutionary viewpoint is the **evolutionarily stable strategy** (ESS; Maynard Smith and Price, 1973).

Definition 13.10

A strategy x is **evolutionarily stable** if, when the whole population plays x, a small group of mutants playing another strategy y cannot invade the population. Formally, for every $y \neq x$

$$u(x, x) > u(y, x) \text{ or } [u(x, x) = u(y, x) \text{ and } u(x, y) > u(y, y)]$$

must hold, where $u(a, b)$ is the payoff of an individual playing a against an opponent playing b.

The two parts of the condition explain why the mutant cannot spread. When mutants are few, almost all their pairings are with resident x-players. The first condition says that in these pairings the mutant earns less than the residents; the mutant quickly recedes. If the first condition holds with equality, what decides is the rare mutant–mutant pairings: the second condition says that when meeting a mutant, the resident earns more than the mutant. Every ESS is a symmetric Nash equilibrium (the inequality of the first condition always holds as $\geq$); but not every Nash equilibrium is an ESS.

The example game is **Hawk–Dove**. Individuals meet over a resource of value $V = 4$. The Hawk always fights; the Dove makes a display but retreats if a fight starts. The cost of fighting is $C = 6$.

- Two hawks fight: one wins (V), one is injured ($-C$); the average is $(V - C)/2 = (4 - 6)/2 = -1$.
- A hawk against a dove takes the resource: $V = 4$; the dove gets 0.
- Two doves share the resource: $V/2 = 2$.

Individual \ Opponent	Hawk	Dove
Hawk	(−1 , −1)	(4 , 0)
Dove	(0 , 4)	(2 , 2)

Figure 13.15: The Hawk–Dove game ($V = 4$, $C = 6$). Row: the individual, column: the opponent; a cell gives the payoffs (individual, opponent).

In the game of Figure 13.15 no pure strategy is an ESS. If the whole population is dove, a hawk mutant gets 4 in every pairing while doves get 2; hawks spread. If the whole population is hawk, a dove mutant gets 0 while hawks get −1; doves spread. At both extremes the minority strategy has the advantage. So the population must settle at a mixture.

13.6.2 Hawk–Dove step by step

Let the hawk share be x and the dove share $1 - x$. A hawk enters a pairing in which its opponent is a hawk with probability x and a dove with probability $1 - x$. The fitnesses with the baseline 3 added:

$$f_H = 3 + (-1) \cdot x + 4 \cdot (1 - x) = 7 - 5x, f_D = 3 + 0 \cdot x + 2 \cdot (1 - x) = 5 - 2x$$

The population average is $\bar{f} = x \cdot f_H + (1 - x) \cdot f_D$, and the new hawk share is $x' = x \cdot f_H / \bar{f}$.

Example 13.17

Let hawks initially be 10% of the population ($x = 0.100$).

1. Fitnesses: $f_H = 7 - 5 \cdot 0.1 = 6.50, f_D = 5 - 2 \cdot 0.1 = 4.80$.
2. Average: $\bar{f} = 0.1 \cdot 6.50 + 0.9 \cdot 4.80 = 0.65 + 4.32 = 4.97$.
3. New share: $x' = 0.1 \cdot 6.50 / 4.97 = 0.65 / 4.97 \approx 0.131$.
4. Since $f_H > \bar{f}$, hawks increased. In the next generation the same calculation is made with $x = 0.131$: $f_H = 6.35, f_D = 4.74, \bar{f} \approx 4.95, x' \approx 0.168$.

generation	x (hawk share)	f_H	f_D	$\bar{f}$	new x = $x \cdot f_H/\bar{f}$
0	0.100	6.50	4.80	4.97	0.131
1	0.131	6.35	4.74	4.95	0.168
2	0.168	6.16	4.66	4.92	0.210
3	0.210	5.95	4.58	4.87	0.257
4	0.257	5.72	4.49	4.80	0.306
5	0.306	5.47	4.39	4.72	0.354
6	0.354	5.23	4.29	4.62	0.401
7	0.401	5.00	4.20	4.52	0.443

Table 13.14: Replicator dynamics in Hawk–Dove ($V = 4$, $C = 6$, baseline 3), $x_0 = 0.1$.

Table 13.14 shows two trends together. The hawk share rises every generation, because $f_H > \bar{f}$ in every row. But the hawks' advantage steadily shrinks: f_H falls from 6.50 to 5.00, and the difference $f_H - f_D$ from 1.70 to 0.80. As hawks multiply, they meet each other more often and get into costly fights. The increase slows down; the population approaches a rest point.

13.6.3 From every starting point to the ESS

The rest point is the share at which the two strategies earn equally; at that point $f_H = f_D = \bar{f}$ and the share does not change:

$$7 - 5x = 5 - 2x \Rightarrow 2 = 3x \Rightarrow x^* = 2/3$$

In general (without the baseline) the equation $(V - C)/2 \cdot x + V(1 - x) = (V/2)(1 - x)$ gives $x^* = V/C$; here $4/6 = 2/3$. This value is the same as the game's **symmetric mixed Nash equilibrium**: in a mixed equilibrium a player must be indifferent between its two strategies, and this is exactly the condition $f_H = f_D$.

The rest point is stable. When $x < 2/3$, $7 - 5x > 5 - 2x$, that is, hawks earn above the average and increase. When $x > 2/3$ the inequality reverses and hawks decrease. From both directions the population is pushed to $2/3$.

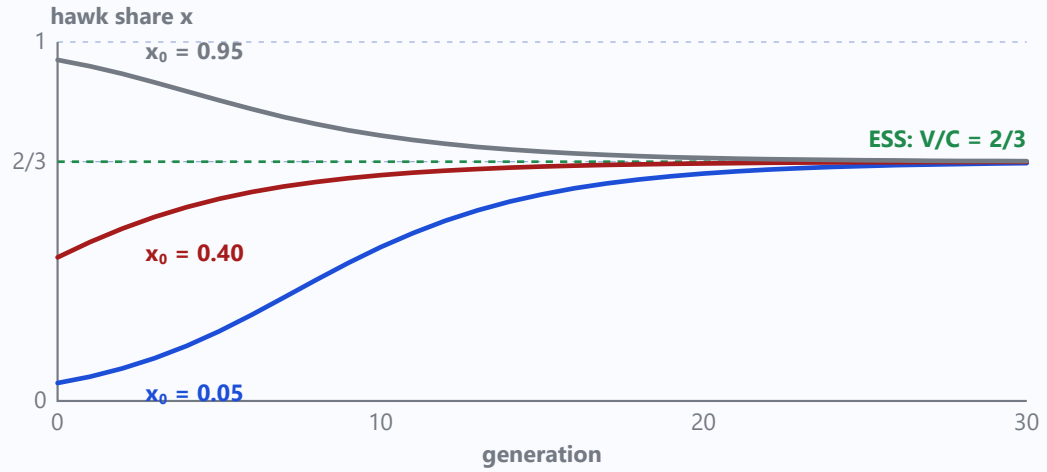


Figure 13.16: Replicator dynamics from three different starting points: $x_0 = 0.05$ (blue), 0.40 (red), 0.95 (gray). Within thirty generations all three curves approach the ESS $x^* = V/C = 2/3$.

Example 13.18

Let us verify that the mixture $x^* = 2/3$ satisfies the ESS condition, using the payoffs without the baseline.

1. **First condition:** against x^* a hawk earns $\tfrac{2}{3} \cdot (-1) + \tfrac{1}{3} \cdot 4 = 2/3$ and a dove earns $\tfrac{2}{3} \cdot 0 + \tfrac{1}{3} \cdot 2 = 2/3$. In a mixed equilibrium every mutant gets the same payoff against x^*: $u(y, x^*) = u(x^*, x^*) = 2/3$. There is equality; the second condition is examined.
2. **Hawk mutant:** $u(x^*, H) = \tfrac{2}{3} \cdot (-1) + \tfrac{1}{3} \cdot 0 = -2/3$, $u(H, H) = -1$. $-2/3 > -1$: satisfied.
3. **Dove mutant:** $u(x^*, D) = \tfrac{2}{3} \cdot 4 + \tfrac{1}{3} \cdot 2 = 10/3$, $u(D, D) = 2$. $10/3 > 2$: satisfied.

The condition holds against both mutants; $x^* = 2/3$ is an ESS.

Figure 13.16 shows that in this game every interior starting point goes to the ESS. Not every game is so well behaved. In **Rock-Paper-Scissors** the replicator dynamics circles around the equilibrium: when rocks multiply, papers gain the advantage; when papers multiply, scissors do; when scissors multiply, rocks do; the population never settles. In the **Stag Hunt**, the dynamics go to one of the two pure equilibria depending on the start: if cooperators (stag hunters) are sufficiently numerous at the start, the stag equilibrium results; if they are few, the hare equilibrium does.

Common mistake

- **Forgetting to multiply the share by fitness and divide:** $x \cdot f_H$ alone is not the new share; without division by $\bar{f}$ the shares do not sum to 1.
- **Mistaking $\bar{f}$ for the simple average of the two fitnesses:** $\bar{f} = x \cdot f_H + (1 - x) \cdot f_D$ is a weighted average, not $(f_H + f_D)/2$.

- **Saying "the hawk wins every pairing, so everyone becomes a hawk":** a hawk wins against a dove, but as hawks multiply, hawk–hawk fights pull the average down. Fitness depends on the composition of the population.

13.7 Repeated games

As a one-shot game, the Prisoner's Dilemma gives one of the bleakest results in game theory: both players defect and both end up worse off than under cooperation. In real life, however, we meet the same people, the same firms, the same neighbors again and again. This section examines how repetition makes cooperation possible, which strategies succeed, and how cooperation spreads in a learning population.

13.7.1 The repeated Prisoner's Dilemma and the grim trigger

Here the Prisoner's Dilemma is written with points; a high score is good. C (staying silent) is cooperation and D is defection.

		Player 2	
		C (cooperate)	D (defect)
Player 1	C (cooperate)	(3 , 3)	(0 , **5**)
	D (defect)	(**5** , 0)	(**1** , **1**)

Figure 13.17: Prisoner's Dilemma payoffs. (C, C) = (3, 3), (C, D) = (0, 5), (D, C) = (5, 0), (D, D) = (1, 1). The marked numbers are the best responses; the only Nash equilibrium is (D, D).

The four numbers in Figure 13.17 have names: the **temptation** 5 (defecting while the opponent cooperates), **mutual cooperation** 3, **mutual defection** 1, and the **sucker's payoff** 0 (cooperating while the opponent defects). The ordering is temptation 5 > mutual cooperation 3 > mutual defection 1 > sucker's payoff 0. Defection is the dominant strategy: if the opponent plays C, D gives 5 and C gives 3; if the opponent plays D, D gives 1 and C gives 0.

A finite and known number of repetitions does not change the outcome. If both players know that the game will last exactly 10 rounds, the last round is a one-shot game and both sides defect. In round 9 the players know what the last round will be; behavior in round 9 does not affect round 10, so round 9 is also like a one-shot game and they defect. This **backward induction** continues to the first round: defection in every round.

The situation changes when it is not known when the game will end. Let the game continue for one more round after each round with probability δ (and end with probability $1 - \delta$). δ is called the **discount factor**; it measures the present value of the next round. This situation is also called the **shadow of the future**: the more probable the future, the longer the shadow it casts over today's behavior. The expected length of the game is $1/(1 - \delta)$ rounds; for example, for $\delta = 0.9$, 10 rounds on average.

Definition 13.11

The grim trigger strategy (Grim): cooperate in the first round; if the opponent defects even once, defect forever from that moment on.

Let us compute whether a deviation by one player is profitable when both players play the grim trigger. The calculation uses a **geometric series**. In the sum $S = a + a\delta + a\delta^2 + \ldots$ each term is δ times the previous one; therefore one can write $S = a + \delta S$ and find $S = a/(1 - \delta)$ (for $0 \le \delta < 1$).

Theorem 13.4 (Cooperation with the grim trigger)

With payoffs temptation 5, mutual cooperation 3 and mutual defection 1, both players playing the grim trigger is an equilibrium if and only if $\delta \ge 1/2$. In general, if the temptation is c, the threshold is $\delta^* = (c - 3)/(c - 1)$.

Proof. **Complying:** if both sides stick to the grim trigger, mutual cooperation is played every round and the player gets 3 per round. The expected total is $3 + 3\delta + 3\delta^2 + \ldots = 3/(1 - \delta)$.

Deviating: if the player defects in this round it gets the temptation 5. Since the opponent plays the grim trigger, it defects in every later round; in those rounds the player's best response is also defection, and it gets 1 from mutual defection. The total is $5 + \delta + \delta^2 + \ldots = 5 + \delta \cdot 1/(1 - \delta) = 5 + \delta/(1 - \delta)$. (Deviating in a later round starts the same calculation from that round; the result does not change.)

Comparison: the condition complying ≥ deviating is $3/(1 - \delta) \ge 5 + \delta/(1 - \delta)$. Multiplying both sides by the positive $(1 - \delta)$ does not reverse the inequality: $3 \ge 5(1 - \delta) + \delta = 5 - 4\delta$. From this, $4\delta \ge 2$, that is, $\delta \ge 1/2 = (5 - 3)/(5 - 1)$. With temptation c the same steps give $3 \ge c(1 - \delta) + \delta$, that is, $\delta(c - 1) \ge c - 3$.

The punishment itself: after a deviation, both sides defecting forever is the repetition of (D, D), the Nash equilibrium of the one-shot game. In the punishment phase it is not profitable for any player to return to C alone (it would get 0 instead of 1); so the threat of punishment is credible. □

Example 13.19

1. ***δ = 0.9*:** complying *3/0.1 = 30*; deviating *5 + 0.9/0.1 = 5 + 9 = 14*. *30 > 14*: cooperation survives.
2. ***δ = 0.3*:** complying *3/0.7 ≈ 4.29*; deviating *5 + 0.3/0.7 ≈ 5 + 0.43 = 5.43*. Deviating is better: defection.
3. **At the threshold, *δ = 0.5*:** complying *3/0.5 = 6*; deviating *5 + 0.5/0.5 = 6*. The two options are equal.

This result is a special case of a much more general fact. The **folk theorem** says: if *δ* is large enough, many average payoffs that give every player more than it can guarantee by defecting (here, more than 1) can be sustained in an equilibrium of the repeated game. The name comes from the fact that nobody knows who first proved the result and that the result circulated among experts like "folk knowledge." The theorem has a good and a bad face. The good face: cooperation is possible. The bad face: there are very many equilibria and the theory does not say which one will be played.

13.7.2 For which δ does cooperation survive?

How the threshold depends on the temptation can be seen by changing the temptation. Complying is worth *3/(1 − δ)* and deviating *temptation + δ/(1 − δ)*; complying ≥ deviating if and only if *δ ≥ δ* = (temptation − 3)/(temptation − 1)*.

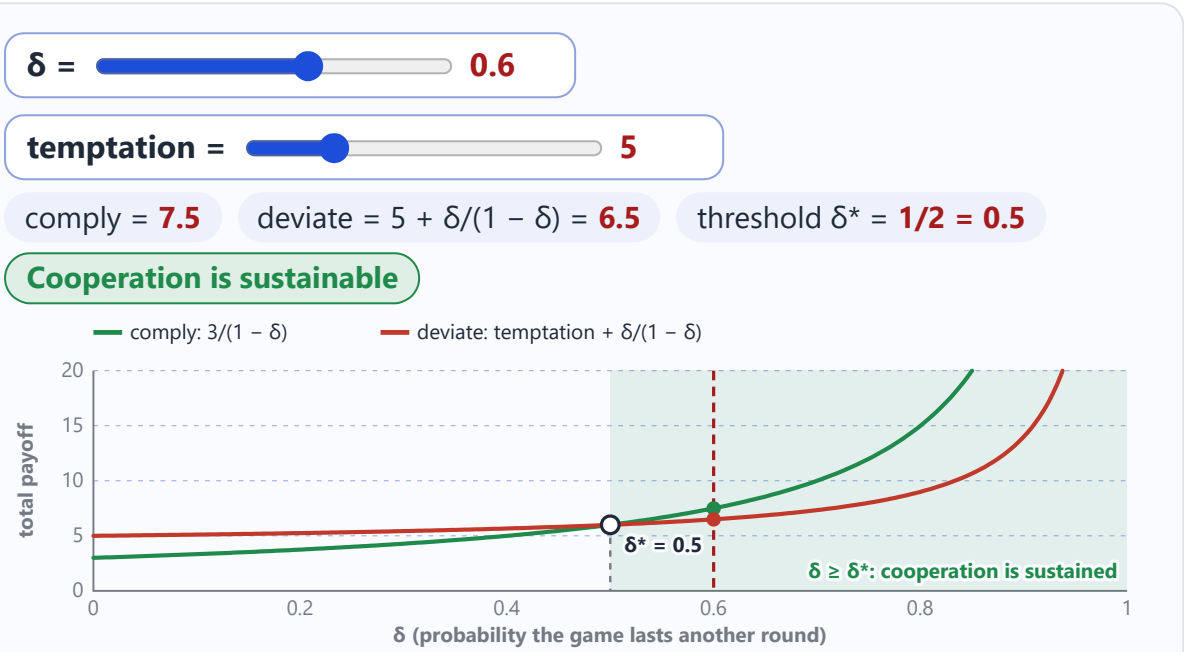


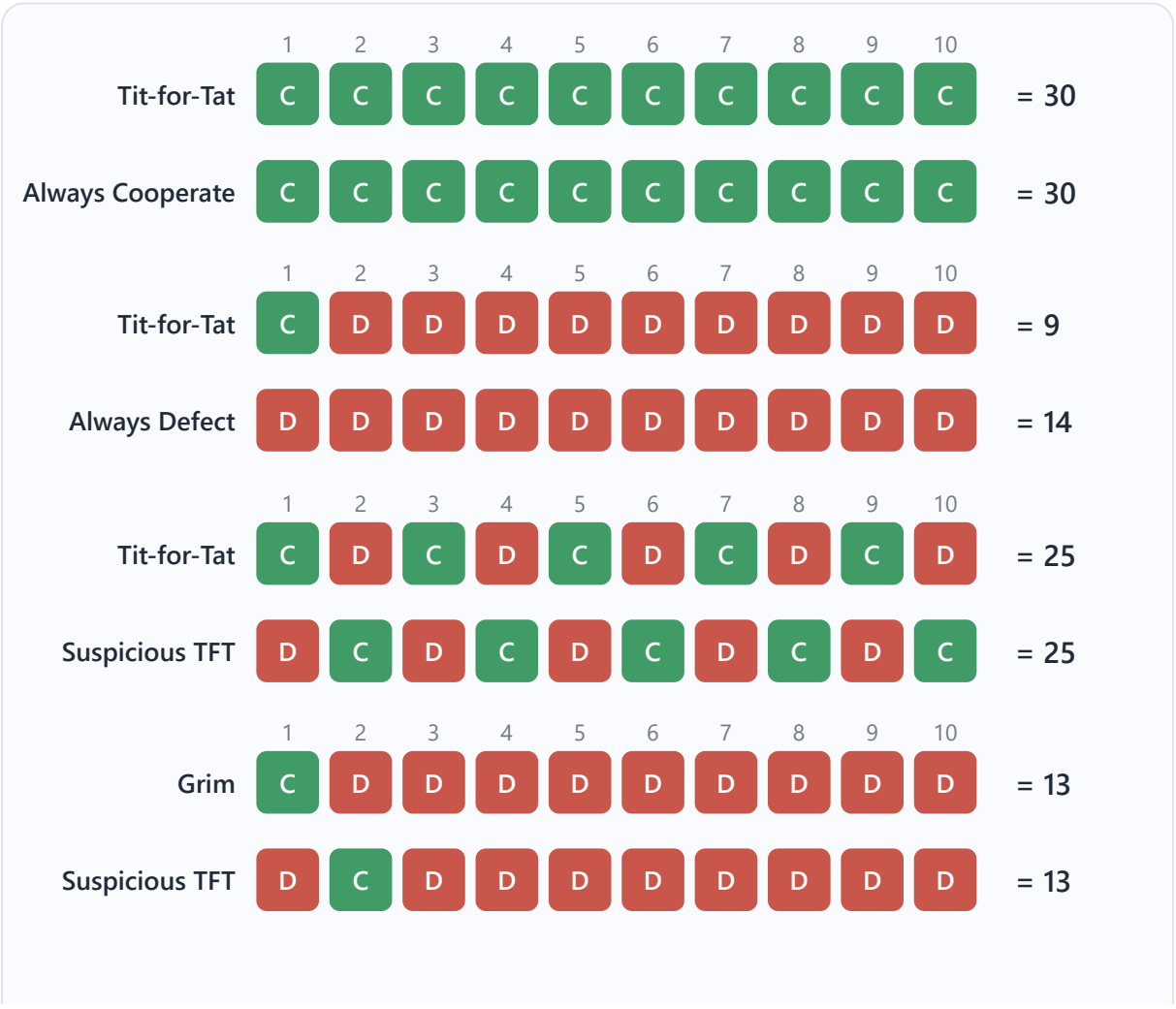
Figure 13.18: The interactive grim trigger chart. The green curve is the value of complying, *3/(1 − δ)*, and the red curve is the value of deviating, *temptation + δ/(1 − δ)*. The point where the two curves cross is the threshold *δ**; in the green region to its right cooperation survives. The sliders change *δ* and the temptation. The default values are *δ = 0.6* and temptation 5: complying 7.5, deviating 6.5, threshold 0.5.

In the chart of Figure 13.18 the two curves start at $\delta = 0$ from 3 and from the temptation value. The deviation curve starts higher but rises more slowly, because a deviating player gets 1 instead of 3 in every future round. The more valuable the future is (the larger δ), the more deterrent the grim trigger threat is. As the temptation grows, the threshold needed for cooperation also rises: with temptation 4, $\delta^* = 1/3$; with temptation 9, $\delta^* = 6/8 = 0.75$. The larger the immediate gain from defection, the more probable the future must be to deter it.

13.7.3 Ten-round matches

The grim trigger preserves cooperation but is very harsh: a single defection is punished forever. A milder and very famous strategy is **Tit-for-Tat** (TFT): it cooperates in the first round and afterward repeats the opponent's last move. For comparison, let us define five strategies:

- **Always Cooperate** (All-C) and **Always Defect** (All-D): always the same move, regardless of the opponent.
- **Tit-for-Tat** (TFT): starts with C, then copies the opponent's last move.
- **Grim**: starts with C; if the opponent defects once, it plays D forever.
- **Suspicious TFT** (STFT): starts with D, then copies the opponent's last move.



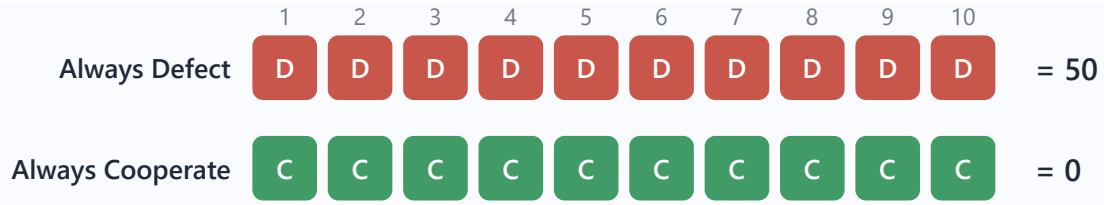


Figure 13.19: Five ten-round matches. Green is C (cooperation), red is D (defection). From top to bottom: TFT – Always Cooperate (30 – 30), TFT – Always Defect (9 – 14), TFT – Suspicious TFT (25 – 25), Grim – Suspicious TFT (13 – 13), Always Defect – Always Cooperate (50 – 0).

The scores of the matches in Figure 13.19 arise as follows:

1. **TFT – Always Cooperate:** TFT answers cooperation with cooperation. Every round is (C, C): both get $10 \cdot 3 = 30$.
2. **TFT – Always Defect:** In round 1 TFT plays C and the opponent D: TFT 0, opponent 5. In the next nine rounds TFT copies the opponent's D and (D, D) continues: 1 each per round. TFT gets $0 + 9 = 9$, the opponent $5 + 9 = 14$. TFT is exploited in the first round and defects afterward.
3. **TFT – Suspicious TFT:** Suspicious TFT starts with defection. The two TFTs "echo" each other: TFT copies the opponent's D while the opponent copies TFT's C; C/D continues alternately. Each player gets 5 in five rounds and 0 in five rounds: both 25.
4. **Grim – Suspicious TFT:** In round 1 Grim plays C and STFT plays D: 0 – 5. In round 2 Grim does not forgive and plays D, while STFT copies Grim's C: 5 – 0. In the next eight rounds both play D: 8 each. Total 13 – 13.
5. **Always Defect – Always Cooperate:** The pure cooperator is completely exploited: $10 \cdot 5 = 50$ to 0.

A remarkable property follows from these matches: **TFT can never win a match**. TFT never defects first and answers each of the opponent's defections with at most one defection; so in a match it cannot collect more points than its opponent. At best it ties. Yet it wins the tournament (13.7.5).

13.7.4 Code trace: match()

The code that plays the matches is short. The `PAY` dictionary maps the combination of two moves (for example `"CD"`) to the two players' scores. The strategies are small functions that take their own history and the opponent's history.

```
PAY = {"CC": (3, 3), "CD": (0, 5),
       "DC": (5, 0), "DD": (1, 1)}
tft = lambda me, opp: opp[-1] if opp else "C"
stft = lambda me, opp: opp[-1] if opp else "D"
def match(s1, s2, T):
    h1, h2, p1, p2 = [], [], 0, 0
    for t in range(T):
        a, b = s1(h1, h2), s2(h2, h1)
        u, v = PAY[a + b]
        h1.append(a); h2.append(b)
        p1, p2 = p1 + u, p2 + v
    return p1, p2
```

- `tft` : if the opponent's history is not empty it returns the opponent's last move (`opp[-1]`), and `"C"` if it is empty. `stft` is the same rule but returns `"D"` on an empty history.
- `match` : in each round the two strategies are called from their own viewpoints. The first player is called with `s1(h1, h2)` and the second with `s2(h2, h1)` ; thus each strategy receives its own history first and then the opponent's.
- `PAY[a + b]` joins the two move letters and finds the score pair. The moves are appended to the histories and the scores are summed.

t	TFT	STFT	Reason	PAY	Scores
0	C	D	empty history: TFT opens with C, STFT with D	CD = (0, 5)	0 – 5
1	D	C	both copy the opponent's last move	DC = (5, 0)	5 – 5
2	C	D	copying	CD = (0, 5)	5 – 10
3	D	C	copying	DC = (5, 0)	10 – 10
4	C	D	copying	CD = (0, 5)	10 – 15
5	D	C	copying	DC = (5, 0)	15 – 15

Table 13.15: Trace of `match(tft, stft, 6)` .



Figure 13.20: Result of `match(tft, stft, 6)` : TFT C D C D C D, Suspicious TFT D C D C D C; scores 15 – 15.

The function returns `(15, 15)` . Figure 13.20 shows an echo cycle: a single initial defection starts a "CD / DC" alternation that goes on forever between two copying strategies. The average score per round is *(0 + 5)/2 = 2.5*; under mutual cooperation it would be 3. The only thing that could break the cycle is one of the two strategies "forgiving" once; TFT and STFT do not do this.

13.7.5 A mini Axelrod tournament

Let the five strategies play a ten-round match against everyone, including their own twin. Table 13.16 gives the score that the strategy in each row collects in ten rounds against the opponent in each column.

Strategy \ opponent	All-C	All-D	TFT	Grim	STFT	Total	Rank
Always Cooperate	30	0	30	30	27	117	2
Always Defect	50	10	14	14	10	98	4
Tit-for-Tat	30	9	30	30	25	124	1
Grim Trigger	30	9	30	30	13	112	3
Suspicious TFT	32	10	25	13	10	90	5

Table 13.16: A mini Axelrod tournament. The score that the row strategy collects in 10 rounds against the column opponent. All-C: Always Cooperate, All-D: Always Defect, STFT: Suspicious TFT.

Example 13.20

Let us compute two cells of the table in detail.

1. **Always Cooperate – Suspicious TFT = 27:** in round 1 All-C plays C and STFT plays D: All-C gets 0. From round 2 on, STFT copies All-C's C and nine rounds of (C, C) are played: $9 \cdot 3 = 27$. The total is $0 + 27 = 27$. In the same match STFT gets $5 + 27 = 32$; this is the first cell of the STFT row.
2. **Always Defect – TFT = 14:** in round 1 All-D gets 5; then nine rounds of (D, D): 9. The total is 14. The same holds against Grim.
3. **Totals:** TFT $30 + 9 + 30 + 30 + 25 = 124$; Always Defect $50 + 10 + 14 + 14 + 10 = 98$.

Ranking: Tit-for-Tat (124), Always Cooperate (117), Grim (112), Always Defect (98), Suspicious TFT (90). **TFT comes first without winning a single match.** Always Defect, in contrast, wins or ties every match but comes second to last. The explanation lies in where the points come from. Always Defect wins individual matches but collects few points in the matches it wins: against cooperators it exploits once, then falls into mutual defection and gets 1 per round. TFT, on the other hand, establishes long mutual cooperation with cooperators and collects 3 per round. The tournament rewards not beating the opponent but the total score.

The tournament table can also be turned into an evolutionary story. Let each strategy initially make up one fifth of the population, and let each strategy's fitness be its average score against an opponent drawn at random from the population: TFT $124/5 = 24.8$, All-C 23.4, Grim 22.4, All-D 19.6, STFT 18.0. The population average is $\bar{f} = 541/25 = 21.64$. With the replicator rule, one generation later the shares are TFT $0.2 \cdot 24.8/21.64 \approx 0.229$, All-C ≈ 0.216, Grim ≈ 0.207, All-D ≈ 0.181, STFT ≈ 0.166. The three "nice" strategies that earn above the average spread; those that start with defection decline.

13.7.6 Axelrod's tournaments and their lessons

The mini tournament is a small copy of a real experiment. The political scientist Robert Axelrod organized two computer tournaments in 1980. Game theorists, mathematicians and computer programmers submitted programs for the repeated Prisoner's Dilemma: 14 programs entered the first tournament and 62 the second. Anatol Rapoport's four-line Tit-for-Tat program won both. The participants in the second tournament knew the results of the first and tried to beat TFT; they still failed (Axelrod, 1984).

Axelrod listed four properties common to successful strategies:

- **Nice:** it never defects first. In the mini tournament the three nice strategies (TFT, All-C, Grim) took the first three places.
- **Retaliatory:** it does not let defection go unpunished. Because Always Cooperate does not retaliate, it got 0 against Always Defect.
- **Forgiving:** if the opponent returns to cooperation, it returns too. Because Grim does not forgive, it got only 13 against Suspicious TFT; TFT got 25.
- **Clear:** the opponent can easily figure out its rule. TFT's rule is understood in a few rounds; the opponent quickly learns that cooperation is best against it.

TFT's weak point is **noise**, that is, the probability that a player mistakenly makes a move it did not intend. In a noisy environment two TFTs enter an endless cycle of retaliation after a single mistake.



Figure 13.21: Two TFTs, a single mistake by A in round 4 (D instead of C). An endless C/D echo follows: 29 – 24 over ten rounds (30 – 30 in the match without the mistake).

In the match of Figure 13.21 the first three rounds are (C, C): 9 points each. In round 4 A mistakenly plays D: A gets 5, B gets 0. In round 5 B copies A's D, while A copies B's C: A gets 0, B gets 5. From then on the two players defect against each other in turn. In the seven rounds from 4 to 10, A gets 5 four times (4, 6, 8, 10) and B three times (5, 7, 9). The totals are *9 + 20 = 29* and *9 + 15 = 24*; the two players' combined total falls from 60 to 53. Even A, who made the mistake, loses out: in the echo cycle it gets 2.5 per round on average, which is less than the 3 of mutual cooperation. As a remedy, **generous Tit-for-Tat** is proposed: it sometimes (with a certain probability) forgives the opponent's defection and breaks the echo cycle; in a noisy environment it does better than TFT. Win-Stay Lose-Shift (WSLS) also returns to cooperation within two rounds after a single mistake.

One last lesson: **success depends on the environment**. No strategy is best in every pool of opponents. If the pool of opponents changes, the ranking changes. For example, if the pool consisted only of Always Defect players, the best strategy would be Always Defect; TFT would be exploited in

the first round of every match. This is why Axelrod also organized evolutionary tournaments: in each generation the strategies multiply in proportion to their success in the previous tournament (replicator dynamics). In these tournaments aggressive strategies multiply first, while there are naive cooperators to exploit; once their victims decline they decline too, and nice, retaliatory strategies such as TFT settle into the population.

Common mistake

- **Saying "TFT wins matches":** TFT cannot win any match; at best it ties. It wins the tournament with a high total score.
- **Forgetting the match against one's own twin:** in the mini tournament every strategy also plays its own twin; the 10 points of Always Defect and the 10 points of STFT come from these matches.
- **Expecting cooperation under finite, known repetition:** the 10 rounds in the tournament are a limit the strategies "do not know"; the programs do not change behavior toward the end. If the number of rounds were known to the players, backward induction would recommend defection in every round.

13.8 Discussion: learning, humans, and ethics

The results of the chapter can be gathered around three discussion questions.

Does learning always lead to the optimum? No. Many counterexamples were seen throughout the chapter. Best-response dynamics cycle in Matching Pennies; fictitious play does not converge in Shapley's 3×3 example; the last iterate of two Hedge players keeps oscillating. The environment and the opponents change too: a learning opponent moves the target. A wrong model also leads to bad convergence: a bandit algorithm that only exploits gets stuck on the wrong arm. For this reason the reliable goal in learning theory is often not "reaching equilibrium" but **no-regret**: the guarantee that, whatever the opponent does, looking back, no fixed strategy would have done much better.

Do humans learn like algorithms? Partly. The similarities are clear: humans also learn by trial and error and change their behavior according to past experience. There are differences too: humans act with risk perception, emotions and social norms, while algorithms follow a defined reward. The finding in the bandit experiment (Daw et al., 2006) showed that humans' exploration behavior resembles softmax choice more than an algorithm such as UCB.

What are the ethical risks in adaptive systems? Learning systems learn the data as it is. Bias in the data is carried into the system's decisions (data bias). A recommender system that learns to maximize click counts may promote content that manipulates the user (clickbait). Learning from user behavior raises privacy problems. A subtler risk is **algorithmic price coordination**: learning bots that set prices in the same market may reach a "tacit agreement" at high prices without any explicit

agreement. This is a form of cooperation in the repeated Prisoner's Dilemma that works against consumers: bots that learn punishment strategies similar to the grim trigger keep the price high by punishing a competitor's discount with a price war.

13.9 Summary

- **In best-response dynamics** the players take turns best-responding to the opponent's last action. If the dynamics stop, they stop at a pure Nash equilibrium; which equilibrium is reached depends on the starting point. In finite **potential games** the dynamics must stop, since every profitable deviation increases the common potential. In games with no pure equilibrium, such as Matching Pennies, they cycle.
- **In fictitious play** the player best-responds to the frequencies of the opponent's entire history. In Matching Pennies the actions go around in ever longer cycles while the frequencies approach (½, ½) (0.472 and 0.492 at 400 rounds). Convergence is guaranteed in zero-sum (Robinson), 2×2 (Miyasawa) and potential games; it fails in Shapley's 3×3 example.
- **External regret** is the total difference between the best fixed action in hindsight and the algorithm: $R_T = \max_i \Sigma\, g_t(i) - \Sigma\, g_t(a_t)$. If $R_T/T \to 0$ the algorithm is no-regret. Deterministic rules such as follow the leader are exploited (regret ≈ $T/2$).
- **Hedge** updates the weights by $w_i \leftarrow w_i \cdot e^{\eta \cdot g_t(i)}$ and plays with probability proportional to the weights. $R_T \le (\ln n)/\eta + \eta T$; with $\eta = \sqrt{(\ln n / T)}$, $R_T \le 2\sqrt{(T \ln n)}$.
- Against a fixed opponent Hedge goes to the best response; against a learning opponent the last iterate cycles but the **time average** approaches equilibrium. If everyone plays no-regret, the empirical distribution of the profiles played approaches the set of **coarse correlated equilibria** (CCE); if internal regret is minimized, it approaches the set of **correlated equilibria** (CE); in zero-sum games the average strategies approach the minimax.
- **In the multi-armed bandit** only the reward of the pulled arm is seen and the exploration–exploitation dilemma arises. **ε-greedy** explores randomly with probability ε; with a fixed ε its regret grows linearly. **UCB1** chooses the arm with the largest score $Q + \sqrt{(2 \ln t / N)}$; it directs exploration toward little-tried arms and its regret grows logarithmically (84 against 118 at 5000 rounds).
- **Q-learning** updates state–action values with $Q(s, a) \leftarrow Q(s, a) + \alpha[r + \gamma \cdot \max Q(s', \cdot) - Q(s, a)]$; reward propagates backward from the goal. In multi-agent settings the environment is non-stationary, so single-agent guarantees no longer hold.
- **In replicator dynamics** a strategy that earns above the average spreads: $x_i' = x_i f_i / \bar{f}$. In Hawk–Dove every interior starting point goes to the **ESS** $x^* = V/C$; this is the symmetric mixed Nash equilibrium.
- **In the repeated Prisoner's Dilemma** the grim trigger preserves cooperation when $\delta \ge (\text{temptation} - 3)/(\text{temptation} - 1)$ ($\delta \ge 1/2$ with payoffs 5, 3, 1, 0). In the mini Axelrod tournament Tit-for-Tat comes first without winning a single match; successful strategies are nice, retaliatory, forgiving and clear. Under noise, generous TFT and WSLS do better.

Method	Description	Result
Best-response dynamics	Best response to the last action	Equilibrium in potential games; otherwise cycles
Fictitious play	Best response to past frequencies	Frequencies converge to equilibrium in zero-sum games
Hedge	$w \leftarrow w \cdot e^{\eta g}$	$R_T = O(\sqrt{(T \ln n)})$; time average → CCE
ε-greedy / UCB	Exploration–exploitation balance	UCB: logarithmic regret
Q-learning	$Q \leftarrow Q + \alpha(r + \gamma \cdot \max Q' - Q)$	Reward propagates backward from the goal
Replicator	Successful strategies spread	Hawk–Dove: $x^* = V/C$
Repeated PD	The shadow of the future	Cooperation if $\delta \geq 1/2$; TFT is strong

Table 13.17: The chapter's methods at a glance.

13.10 Key terms

best-response dynamics potential game fictitious play external regret internal / swap regret follow the leader multiplicative weights learning rate correlated equilibrium coarse correlated equilibrium multi-armed bandit exploration–exploitation ε-greedy upper confidence bound Thompson sampling adversarial bandit reinforcement learning Q-learning discount factor non-stationary counterfactual regret minimization replicator dynamics fitness evolutionarily stable strategy grim trigger strategy folk theorem Tit-for-Tat suspicious / generous Tit-for-Tat win-stay lose-shift

13.11 Test Yourself

1. In the cells (u_1, u_2): (Top, Left) = (2, 1), (Top, Right) = (0, 0), (Bottom, Left) = (0, 0), (Bottom, Right) = (1, 2). Where do best-response dynamics, in which the row player moves first and then the column player, stop if they start at (Top, Right)? If they start at (Bottom, Left)? Is this game a

potential game?

Answer

From (Top, Right): against Right the row player sees Top 0, Bottom 1 and switches to Bottom: (Bottom, Right). Against Bottom the column player sees Left 0, Right 2; it stays at Right. The dynamics stop at (Bottom, Right). From (Bottom, Left): against Left the row player sees Top 2, Bottom 0 and switches to Top: (Top, Left). Against Top the column player sees Left 1, Right 0; it stays. The dynamics stop at (Top, Left). The equilibrium reached depends on the start. The game is a potential game: if we choose $\Phi(\text{Top, Left}) = 2$, $\Phi(\text{Top, Right}) = 1$, $\Phi(\text{Bottom, Left}) = 0$, $\Phi(\text{Bottom, Right}) = 2$, then in every unilateral change the payoff difference of the player who changes equals the difference in Φ. For example, in the Right column, when the row player moves from Top to Bottom its payoff rises from 0 to 1 and Φ rises from 1 to 2; in the Top row, when the column player moves from Right to Left its payoff rises from 0 to 1 and Φ rises from 1 to 2.

2. In Hedge with two actions, $\eta = \ln 3$ and the initial weights are 1. The payoffs of the three rounds are $(1, 0)$, $(0, 1)$, $(1, 0)$. What are the probabilities in each round, the algorithm's total expected payoff, and the external regret?

Answer

$e^{\ln 3} = 3$: the weight of an action with payoff 1 triples. Round 1: $w = (1, 1)$, $p = (\tfrac{1}{2}, \tfrac{1}{2})$, expected payoff 0.5; update $w = (3, 1)$. Round 2: $p = (3/4, 1/4)$, payoff $\tfrac{1}{4} \cdot 1 = 0.25$; $w = (3, 3)$. Round 3: $p = (\tfrac{1}{2}, \tfrac{1}{2})$, payoff 0.5; final weights $(9, 3)$. Total $0.5 + 0.25 + 0.5 = 1.25$. Fixed actions: A = 2, B = 1. Regret $2 - 1.25 = 0.75$. In round 2 A was trusted too much and B turned out to win.

3. In a problem with four actions ($n = 4$), Hedge will be played for 400 rounds. What are the η that minimizes the bound, the regret bound, and the per-round regret bound? How do these numbers change if the number of rounds is multiplied by four (1600 rounds)?

Answer

$\ln 4 \approx 1.386$. $\eta = \sqrt{(1.386/400)} \approx 0.059$. The bound is $2\sqrt{(400 \cdot 1.386)} = 2\sqrt{554.5} \approx 47.1$; per round $47.1/400 \approx 0.118$. At 1600 rounds η is halved (≈ 0.029), the total bound is multiplied by $\sqrt{4} = 2$ (≈ 94.2), and the per-round bound is halved (≈ 0.059).

4. In a two-armed bandit the true means are $\mu = (0.6, 0.4)$. In the first five rounds arm 0 has been pulled four times ($Q = 0.8$) and arm 1 once ($Q = 0.5$). (a) Which arm does UCB1 choose at $t = 6$? (b)

If after the choice the pull counts are $N = (4, 2)$, what is the regret over six rounds?

Answer

(a) $2 \ln 6 \approx 3.58$. Arm 0: $0.8 + \sqrt{(3.58/4)} = 0.8 + 0.95 \approx 1.75$. Arm 1: $0.5 + \sqrt{3.58} = 0.5 + 1.89 \approx 2.39$. UCB1 chooses arm 1: its estimate is lower but its uncertainty is large. (b) $R = 6 \cdot 0.6 - (4 \cdot 0.6 + 2 \cdot 0.4) = 3.6 - 3.2 = 0.4$. Regret is computed with the true means; the observed Q values are not used.

5. In Q-learning, $Q(s, a) = 0.5$, $r = 2$, $\gamma = 0.5$, $\max Q(s', \cdot) = 3$, $\alpha = 0.2$. What are the target and the new $Q(s, a)$? What would the result be if s' were a goal state (the episode ended)?

Answer

The target is $2 + 0.5 \cdot 3 = 3.5$. The new value is $0.5 + 0.2 \cdot (3.5 - 0.5) = 0.5 + 0.6 = 1.1$. If s' were a goal, the future value would be taken as 0: the target is 2 and the new value $0.5 + 0.2 \cdot 1.5 = 0.8$.

6. In Hawk–Dove, $V = 2$, $C = 8$, and the baseline is 3. What are the fitnesses when the hawk share is x? What is the ESS? What is the hawk share one generation after $x = 0.5$?

Answer

Payoffs: HH $(2 - 8)/2 = -3$, HD 2, DH 0, DD 1. $f_H = 3 - 3x + 2(1 - x) = 5 - 5x$, $f_D = 3 + 1 \cdot (1 - x) = 4 - x$. The rest point is $5 - 5x = 4 - x \Rightarrow x^* = 1/4 = V/C$. At $x = 0.5$, $f_H = 2.5$, $f_D = 3.5$, $\bar{f} = 3$; $x' = 0.5 \cdot 2.5/3 \approx 0.417$. Since hawks earn below the average they decrease, falling toward 0.25. When fighting is costly, the hawk share is small.

7. In a repeated Prisoner's Dilemma the payoffs are temptation 7, mutual cooperation 4, mutual defection 2, sucker's payoff 0. What is the smallest δ for which cooperation with the grim trigger survives?

Answer

Complying $4/(1 - \delta)$, deviating $7 + 2\delta/(1 - \delta)$. Multiplying by $(1 - \delta)$ gives $4 \geq 7(1 - \delta) + 2\delta = 7 - 5\delta$, that is, $5\delta \geq 3$ and $\delta \geq 3/5$. The general rule: $\delta^* = (\text{temptation} - \text{cooperation})/(\text{temptation} - \text{defection}) = (7 - 4)/(7 - 2)$. Because the payoff of mutual defection is higher, the punishment is milder; the threshold adjusts accordingly.

8. Win-Stay Lose-Shift (WSLS; it plays C in the first round, repeats the same move if it got 3 or 5, and changes its move if it got 0 or 1) plays against Always Defect for six rounds. What are the sequence

of moves and the scores? What would TFT get in the same match?

Answer

Round 1: WSLS C, opponent D: 0 – 5; since WSLS got 0 it changes. Round 2: D – D: 1 – 1; since it got 1 it changes again. The sequence is C D C D C D: WSLS gets $0 + 1 + 0 + 1 + 0 + 1 = 3$, Always Defect gets $5 + 1 + 5 + 1 + 5 + 1 = 18$. After the first round TFT would defect every round: $0 + 5 \cdot 1 = 5$ against $5 + 5 = 10$. WSLS is more robust to noise than TFT, but against an unconditional defector it is exploited every other round.

13.12 Further reading

- T. Roughgarden, *Twenty Lectures on Algorithmic Game Theory* (2016), Lectures 16–18: no-regret learning, the proof of Hedge's regret bound, correlated equilibria, and the learning-based proof of the minimax theorem, presented from a computer science viewpoint.
- T. Lattimore and C. Szepesvári, *Bandit Algorithms* (2020): a comprehensive source giving detailed analyses of ε-greedy, UCB, Thompson sampling and EXP3.
- R. Axelrod, *The Evolution of Cooperation* (1984): tells the story of the computer tournaments, the success of Tit-for-Tat and the evolutionary tournaments without requiring mathematical background.
- D. Easley and J. Kleinberg, *Networks, Crowds, and Markets* (2010), Chapter 7: presents evolutionary game theory, the ESS and Hawk–Dove-like examples at an introductory level.

13.13 References

1. P. Auer, N. Cesa-Bianchi, P. Fischer, “Finite-time analysis of the multiarmed bandit problem”, *Machine Learning* 47, 2002.
2. P. Auer, N. Cesa-Bianchi, Y. Freund, R. E. Schapire, “The nonstochastic multiarmed bandit problem”, *SIAM Journal on Computing* 32(1), 2002.
3. R. Axelrod, *The Evolution of Cooperation*, Basic Books, 1984.
4. A. Blum, Y. Mansour, “From external to internal regret”, *JMLR* 8, 2007.
5. N. Cesa-Bianchi, G. Lugosi, *Prediction, Learning, and Games*, Cambridge, 2006.
6. N. D. Daw, J. P. O’Doherty, P. Dayan, B. Seymour, R. J. Dolan, “Cortical substrates for exploratory decisions in humans”, *Nature* 441, 2006.
7. D. Easley, J. Kleinberg, *Networks, Crowds, and Markets*, Cambridge, 2010 (Chapter 7: evolutionary game theory).
8. D. Foster, R. Vohra, “Calibrated learning and correlated equilibrium”, *Games and Economic Behavior* 21, 1997.
9. D. Fudenberg, D. Levine, *The Theory of Learning in Games*, MIT Press, 1998.

10. S. Hart, A. Mas-Colell, “A simple adaptive procedure leading to correlated equilibrium”, *Econometrica* 68(5), 2000.

11. J. Hofbauer, K. Sigmund, *Evolutionary Games and Population Dynamics*, Cambridge, 1998.

12. A. R. Karlin, Y. Peres, *Game Theory, Alive*, AMS, 2017 (Chapter 7: evolutionary equilibrium; Chapter 18: adaptive decision making).

13. T. Lattimore, C. Szepesvári, *Bandit Algorithms*, Cambridge, 2020.

14. J. Maynard Smith, G. R. Price, “The logic of animal conflict”, *Nature* 246, 1973.

15. K. Miyasawa, “On the convergence of the learning process in a 2×2 non-zero-sum two-person game”, Princeton Research Memo 33, 1961.

16. V. Mnih et al., “Human-level control through deep reinforcement learning”, *Nature* 518, 2015.

17. D. Monderer, L. S. Shapley, “Potential games”, *Games and Economic Behavior* 14(1), 1996.

18. N. Nisan, T. Roughgarden, É. Tardos, V. Vazirani (eds.), *Algorithmic Game Theory*, Cambridge, 2007 (Chapter 4: learning, regret, and equilibria).

19. M. Nowak, K. Sigmund, “A strategy of win-stay, lose-shift that outperforms tit-for-tat in the Prisoner’s Dilemma game”, *Nature* 364, 1993.

20. J. Robinson, “An iterative method of solving a game”, *Annals of Mathematics* 54(2), 1951.

21. T. Roughgarden, *Twenty Lectures on Algorithmic Game Theory*, Cambridge, 2016 (Lectures 16–18).

22. L. S. Shapley, “Some topics in two-person games”, *Advances in Game Theory*, 1964.

23. W. R. Thompson, “On the likelihood that one unknown probability exceeds another in view of the evidence of two samples”, *Biometrika* 25, 1933.

24. C. Watkins, *Learning from Delayed Rewards*, PhD thesis, University of Cambridge, 1989.

25. C. Watkins, P. Dayan, “Q-learning”, *Machine Learning* 8, 1992.

26. M. Zinkevich, M. Johanson, M. Bowling, C. Piccione, “Regret minimization in games with incomplete information”, *NeurIPS*, 2007.

Future Directions

The earlier chapters of the course studied games, equilibria, and mechanisms mostly among people, under fixed rules, and at sizes that can be solved by hand. Today the same questions are being asked again in a new setting. Some of the players are now software agents that speak on behalf of a user. Mechanisms are learned from data instead of being designed by hand. Some rules are enforced by public code that no one can change afterwards. This chapter uses worked numerical examples to show how the tools of the course apply in these three areas and where they fall short: bargaining between agents and algorithmic collusion, learning prices from data and privacy, transaction fee mechanisms and the power to order transactions, and finally the problems of computation at scale. The experimental protocol is summarized in 14.2.8; the table in 14.6 links to the map that ties all the chapters of the course together.

LEARNING OUTCOMES

New players
Describe the game-theoretic questions that AI agents raise in bargaining and pricing.

Design from data
Explain learning a mechanism from data and the notion of sample complexity with a numerical example.

On-chain mechanisms
Interpret transaction fee auctions, the base fee rule, and MEV (maximal extractable value, the value obtainable from the power to order transactions).

Tying the course together
Relate all the chapters of the course on a single map.

14.1 Introduction

This chapter builds no new theory; it carries the tools learned in earlier chapters into new settings. Backward induction, trigger strategies in repeated games, incentive compatibility, revenue maximization, the second-price idea, voting rules, and the price of anarchy all reappear here. What changes is the nature of the players and of the rules. For this reason every section asks two questions together: what does the tool of the course say in this case, and which of its assumptions no longer holds?

The chapter touches on fast-moving areas. Particular products, versions, and current practices change within a few years. The notes therefore tie the examples to lasting principles rather than to product names: patience is bargaining power, anything selected and measured on the same data has its success overstated, and an ordering power that everyone can see carries value. These principles remain valid even when the form of the application changes.

14.1.1 Starting with a game: the AI agent tournament

The chapter opens with a small tournament. Four simple agents play a chosen game against one another. Each match lasts 60 or 120 rounds. In the full tournament every agent meets every other agent once; for four agents that means $4 \cdot 3 / 2 = 6$ matches. The ranking is based on the total payoff each agent collects over all its matches.

The agents are:

- **Random:** in every round it picks one of the two moves with probability one half.
- **Always Cooperate:** it cooperates in every round and never looks at its opponent.
- **Always Defect:** it defects in every round.
- **Best Response:** it predicts that the opponent's next move will be the one the opponent has played most often so far, and plays the best response to it. When there is no history, that is, in the first round, it cooperates.



Figure 14.1: The pairing table of the tournament. At the first level, the Prisoner's Dilemma, every match lasts 60 rounds; each agent plays each of the other three once, for 6 matches in total. Cells with a question mark are matches still to be played.

Figure 14.1 shows the pairings. The tournament raises two intriguing questions. First, which of these four agents finishes first? Second, does the agent that wins one game also win another? The answer to the first can be computed from the structure of the game; the second is one of the main lessons of the chapter. Both questions rest on the notions of repeated games and best response, and both are answered with numbers in 14.2.7.

The importance of this tournament does not come from whether the agents are "smart". The agents are very simple. What matters is that an agent's success depends not only on its own rule but also on its opponents and on the game being played. This is the basic observation of game theory, and it gains weight as software agents become widespread.

14.1.2 A field in motion

Algorithmic game theory arose at the intersection of computer science and economics, and its applications are growing quickly. In industry and academia it is used in areas such as autonomous vehicles, online platforms, and energy networks. Three technological trends fuel this growth:

- **Artificial intelligence and machine learning:** some of the decision makers are now learning software.
- **Big data and real-time decisions:** prices, bids, and routes are updated within seconds.
- **Blockchains and smart contracts:** rules are enforced automatically by public code.

Four research axes arise from these trends. Each is directly related to a chapter of the course.

Axis	Question	Counterpart in the course
Fairness and ethics	Can mechanisms be fair and transparent?	Fairness criteria, fair division
Dynamic and learning systems	How do adaptive autonomous agents behave?	Learning dynamics, regret
Large networks	How can strategic interaction among millions of players be analyzed?	Congestion games, price of anarchy
Incentives and policy	Which rules should be designed for social welfare and sustainability?	Mechanism design, pricing of externalities

Table 14.1: The four axes of the field and their counterparts in the course.

The aim of this chapter is to show how the tools of the course are used in these new areas and where they fall short. Falling short is instructive too. For example, classical mechanism design assumes that the designer will stick to the announced rule afterwards. Smart contracts enforce this assumption in code but create a new problem: pending transactions are visible to everyone, and this visibility can be exploited. Similarly, classical theory assumes that players are rational. Agents based on language models, by contrast, can be sensitive to how a problem is framed and may deviate from this assumption.

14.1.3 Application areas

The four areas where game theory is used most today can be summarized as follows.

Finance
: Algorithmic trading and portfolio management. Trading algorithms play against one another: the profit of one algorithm depends on the orders and the speed of the others. This is a game, not a single-agent optimization.

Traffic and logistics
: Autonomous vehicle networks and route choice. Each vehicle wants to shorten its own travel time; this selfish routing can lead to outcomes such as Braess's paradox, where adding a road to the network slows everyone down.

Online platforms
: Recommender systems, ad auctions, and social network analysis. The ad slots next to search results are sold in a separate auction for each search.

Energy grids
: Load balancing, demand response, and electricity markets. Consumers shift their consumption in response to price signals; producers bid into the market.

What these areas have in common is that the decision makers are increasingly software. A trading algorithm, a routing app, or an ad-bidding tool is an agent playing on behalf of its owner. The rest of the chapter starts from this observation.

14.2 AI agents

This section examines the questions that arise once software agents appear as players: the new problems agents create, bargaining between agents (finite and infinite horizon), and collusion among pricing algorithms. The results of the opening tournament are also explained with these tools.

14.2.1 Strategic AI agents

Agents based on large language models can take on tasks such as shopping, bargaining, planning, and resource sharing on a user's behalf. The agent is then no longer merely a tool; it is an actor with a goal, knowledge, and a strategy. In other words, it is a **player** in the game-theoretic sense. One can ask what the player wants, what it knows, and what it can choose. This brings the old questions of game theory back onto the agenda.

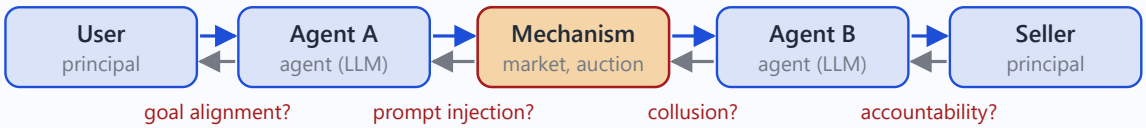


Figure 14.2: The structure in which two agents meet in a mechanism (a market or an auction). The user and the seller are the principals; the agents are their representatives. The red labels show the risk on each link: goal alignment, prompt injection, collusion, and accountability.

Figure 14.2 shows the typical structure. A user assigns a task to agent A, and a seller assigns one to agent B. The two agents meet in a mechanism, for example in a market or an auction. Every link of this chain raises a separate game-theoretic question.

Definition 14.1

Principal–agent problem: One party (the principal) delegates a task to another party (the agent). If the agent's goal differs from the principal's and the principal cannot observe every action of the agent, the agent may act in the interest of its own goal rather than the principal's. The principal's problem is to set up an incentive or monitoring scheme that aligns the agent's goal with its own.

This problem is much older than AI agents. The relationships between a homeowner and a real estate agent, a shareholder and a manager, or a patient and a doctor have the same structure. For example, a commission-based real estate agent may not want to spend weeks of effort to sell the house at a slightly higher price, because only a small part of the price increase comes to the agent, while the whole effort falls on the agent. With software agents the problem appears in four forms.

Goal alignment

Is the agent's goal the same as the user's? The agent's goal is inferred from the user's words or from training data. A user says "find the cheapest" but may in fact also care about fast delivery. If the agent follows the stated goal literally, it can miss the user's real utility. This is the most direct form of the principal–agent problem.

The rationality assumption

Classical game theory assumes that players maximize their payoffs. The behavior of an agent based on a language model, however, can be sensitive to how the problem is presented, that is, to **framing**. It may choose different moves in two different descriptions of the same game. For this reason the Nash equilibrium is not always a good prediction of the behavior of these agents. The observations made about people in behavioral game theory apply here to agents as well.

Manipulation

The agent reads text written by the other side. The opposing agent can try to steer the agent in its own favor by embedding hidden instructions in its message. This is called **prompt injection**. From the game-theoretic point of view, it is a new kind of strategy in which the other side can influence not only the choice of moves but also the opponent's decision rule.

Identity and accountability

On whose behalf is the agent speaking, who is responsible if something goes wrong, and how is an agreement that has been made verified? Mechanism design assumes that it is known who the players are and that a contract is enforceable. These assumptions must be established separately for agents.

Taken together, these four questions say the following: agents do not make game theory unnecessary; on the contrary, they make its questions more important. Which game the agent is playing, on whose behalf it plays, and what the rules of the game are must all be modeled explicitly. Bargaining and pricing models are two concrete examples of this modeling.

14.2.2 Bargaining between agents: finite horizon

Suppose two agents must share an amount. For example, a buyer agent and a seller agent bargain over how to split a gain of 100 TL arising from a transaction. Bargaining proceeds through proposals made in turn.

Definition 14.2

Alternating-offers bargaining: Two players share an amount. In odd-numbered rounds A, and in even-numbered rounds B, proposes a division. If the other side accepts the proposal, the game ends and the division is carried out. If it rejects, play moves on to the next round. Every delay multiplies the value to be shared by a **discount factor** δ. In a game of T rounds, if the last proposal is also rejected, both sides get 0.

$\delta = 0.9$ means that the value shrinks by 10% in every round. This has two interpretations. The first is that there is a real loss: for example, a product that spoils or an opportunity that slips away. The second is that the players are impatient: 100 TL received one round later is worth as much as 90 TL received today. In both interpretations, the larger δ is, the more patient the player.

Because the game is finite, it is solved by **backward induction**. We start from the last round and apply the rule "the proposer offers the smallest share that leaves the other side indifferent between accepting and rejecting" in every round. Whatever the other side would get in the next round if it rejects, the present value of that amount is its lower bound.

Example 14.1

100 TL is to be shared, $\delta = 0.9$, and the game lasts four rounds. A proposes in the odd rounds (1 and 3) and B in the even rounds (2 and 4).

1. **Round 4 (B proposes, last round).** If A rejects, both get 0. For A there is no difference between accepting and rejecting; the indifferent side is assumed to accept. B proposes 100 for itself and 0 for A: $(A, B) = (0, 100)$.
2. **Round 3 (A proposes).** If B rejects, it gets 100 in round 4. Its present value is $0.9 \cdot 100 = 90$. A must offer B at least 90; offering more is unnecessary. The division is $(10, 90)$.
3. **Round 2 (B proposes).** If A rejects, it gets 10 in round 3; the present value is $0.9 \cdot 10 = 9$. B offers A 9: $(9, 91)$.
4. **Round 1 (A proposes).** If B rejects, it gets 91 in round 2; the present value is $0.9 \cdot 91 = 81.9$. A offers B 81.9 and keeps $100 - 81.9 = 18.1$ for itself: $(18.1, 81.9)$.

B accepts the round 1 proposal immediately; the bargaining ends without delay and no value is lost. This outcome is a **subgame perfect equilibrium**: in every round, even in the game starting from that round, everyone plays a best move.



Figure 14.3: Backward induction in four-round bargaining. At each step from right to left, the other side's waiting value is multiplied by 0.9. In round 4 B takes everything; in round 1, in equilibrium, A gets 18.1 and B gets 81.9, and the proposal is accepted immediately.

Figure 14.3 reads the solution from right to left. There is a surprising result at first sight: although A makes the first proposal, it gets only 18.1% of the pie. The reason is that B has the last word. In round 4 B can take everything, and this threat propagates back through all the rounds. With each round we go back, the advantage of the last word shrinks by a factor of δ but does not vanish entirely.

The same calculation can be carried out for different numbers of rounds. One observation shortens the work. Let x_T be the share of the first proposer (A) in a game of T rounds. In the first round, A must satisfy the other side up to its waiting value. For B, the game starting from the second round is a game of $T - 1$ rounds in which B itself is the first proposer; there B gets the share x_{T-1}. Therefore:

$$x_1 = 100, \quad x_T = 100 - \delta \cdot x_{T-1}$$

The rule starts with the one-round game: with a single round, A proposes everything and B accepts (rejection gives both 0). Applying the rule repeatedly yields Table 14.2.

Number of rounds T	1	2	3	4	5	6	7	8	∞
A's share	100	10	91	18.1	83.7	24.7	77.8	30	52.6

Table 14.2: The equilibrium share of the first proposer (A) by number of rounds, for 100 TL and $\delta = 0.9$.

In the table each value is computed from the previous one. For example, $x_2 = 100 - 0.9 \cdot 100 = 10$, $x_3 = 100 - 0.9 \cdot 10 = 91$, $x_4 = 100 - 0.9 \cdot 91 = 18.1$, $x_5 = 100 - 0.9 \cdot 18.1 = 83.71$. The values oscillate: if the number of rounds is odd, the last word is A's and A gets a large share; if it is even, the last word is B's and A gets a small share. The case $T = 1$ is the **ultimatum game**: there is a single proposal, and the other side either accepts or both get 0. The theoretical prediction is that the proposer takes almost everything.

The amplitude of the oscillation shrinks as the number of rounds grows, and the values approach about 52.6. This limit is derived in 14.2.3. Two lessons follow for agent design. The first is that **patience** (the size of δ) and **the right of the last word** are bargaining power. The second is that **knowledge of the horizon is also strategic**, because the deadline determines who makes the last offer. An agent gains power by knowing when the bargaining will end, or by being able to determine it.

Common mistake

- **Assuming the first proposer is always strong:** With a finite horizon, the power lies with the last proposer. At $T = 4$, the first proposer A gets only 18.1.
- **Writing the waiting value without discounting:** B's lower bound in round 3 is not 100 but $0.9 \cdot 100 = 90$; 100 one round later is worth 90 today.
- **Starting backward induction from the beginning:** The calculation starts from the last round. Without knowing what will happen in the last round, the proposal in the first round cannot be found.

14.2.3 Infinite horizon: the Rubinstein solution

If bargaining has no last round, backward induction cannot start; there is no final round to start from. Rubinstein (1982) showed that even in this case there is a unique subgame perfect equilibrium. The solution is found with **stationary strategies**. A stationary strategy makes the same offer in every round, without looking at the past. When A proposes, it always keeps the share x^* for itself; when B proposes, it always leaves A the share y^*. Let us write the shares as fractions of 1 (the whole amount).

The equilibrium has two conditions.

1. **B is indifferent about A's proposal.** A's proposal leaves B $1 - x^*$. If B rejects, it proposes in the next round and gets $1 - y^*$; the present value of that is $\delta_B(1 - y^*)$. A leaves B exactly this much: $1 - x^* = \delta_B \cdot (1 - y^*)$.
2. **A is indifferent about B's proposal.** B's proposal leaves A y^*. If A rejects, it gets x^* in the next round; the present value is $\delta_A x^*$. Hence $y^* = \delta_A \cdot x^*$.

Substitute the second equation into the first: $1 - x^* = \delta_B(1 - \delta_A x^*) = \delta_B - \delta_A\delta_B x^*$. Collect the terms containing x^* on one side: $x^* - \delta_A\delta_B x^* = 1 - \delta_B$, that is, $x^*(1 - \delta_A\delta_B) = 1 - \delta_B$. Divide both sides by $1 - \delta_A\delta_B$.

Theorem 14.1 (Rubinstein, 1982)

In infinite-horizon alternating-offers bargaining with discount factors δ_A, $\delta_B < 1$, there is a unique subgame perfect equilibrium. The share of the first proposer A is

$$x^* = (1 - \delta_B) / (1 - \delta_A\delta_B)$$

and the proposal is accepted in the first round. With symmetric patience ($\delta_A = \delta_B = \delta$), $x^* = 1/(1 + \delta)$.

The symmetric formula follows from the general one: since $1 - \delta^2 = (1 - \delta)(1 + \delta)$, we get $(1 - \delta)/(1 - \delta^2) = 1/(1 + \delta)$. For $\delta = 0.9$ this gives $100/1.9 \approx 52.6$ TL. This is the value toward which the oscillation in Table 14.2 converges. The connection is clear: the fixed point of the finite-horizon rule $x_T = 100 - \delta x_{T-1}$, that is, the solution of the equation $x = 100 - \delta x$, is $x = 100/(1 + \delta)$. As the horizon lengthens, the effect of the last round is multiplied by δ at every step and dies out, and the share approaches this fixed point.

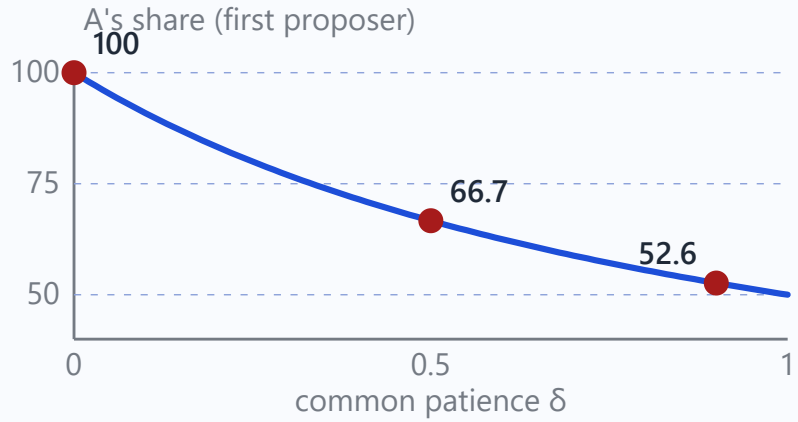


Figure 14.4: The first proposer's share $1/(1+\delta)$ under symmetric patience. With no patience at all ($\delta = 0$) the proposer takes everything; at $\delta = 0.5$ it gets 66.7, and at $\delta = 0.9$ it gets 52.6. As δ approaches 1, the share falls to one half.

Figure 14.4 shows the advantage of the first proposer. Among impatient players (small δ) the first proposal is very valuable, because if the other side rejects, waiting for the next round is costly for it. As patience grows, waiting becomes cheap and the first proposer's advantage shrinks. Very patient players share roughly equally.

When patience differs, the result is more striking. Table 14.3 shows the shares the formula gives in four examples.

δ_A	δ_B	Calculation	A	B
0.9	0.9	0.1 / 0.19	52.6	47.4
0.9	0.8	0.2 / 0.28	71.4	28.6
0.8	0.9	0.1 / 0.28	35.7	64.3
0.99	0.9	0.1 / 0.109	91.7	8.3

Table 14.3: The Rubinstein division (100 TL). A is the first proposer.

Example 14.2

Let $\delta_A = 0.9$ and $\delta_B = 0.8$. A's share is found as follows.

1. Numerator of the fraction: $1 - \delta_B = 1 - 0.8 = 0.2$.
2. Denominator of the fraction: $\delta_A\delta_B = 0.72$, so $1 - 0.72 = 0.28$.
3. $x^* = 0.2 / 0.28 \approx 0.714$: A gets 71.4 TL and B gets 28.6 TL.
4. Check: in B's proposal A is left $y^* = \delta_A\, x^* = 0.9 \cdot 0.714 \approx 0.643$. B must be indifferent between accepting and rejecting A's proposal: $\delta_B(1 - y^*) = 0.8 \cdot 0.357 \approx 0.286 = 1 - x^*$ ✓.

If the roles stay the same but the patience levels are swapped ($\delta_A = 0.8$, $\delta_B = 0.9$), A gets only 35.7 even though it is the first proposer. The difference in patience is stronger than the first-proposer advantage.

The last row of the table is an extreme case: $\delta_A = 0.99$ represents a very patient A, and A gets 91.7% of the pie. The lesson for agent design is this: **patience is power**. An agent that is truly hasty (with a low δ) gets a smaller share. With incomplete information the situation is somewhat subtler: if the other side does not know the agent's true patience, *appearing* hasty also has a similar effect by changing the opponent's beliefs. It is therefore important that a bargaining agent not leak the user's deadline or urgency to the other side.

Common mistake

Confusing finite and infinite horizons. With a finite horizon the last proposer is strong: at $T = 4$, A gets only 18.1. With an infinite horizon there is no last round and the first proposer is slightly ahead: 52.6. Whether a bargaining process has a deadline completely changes who is strong.

14.2.4 Can pricing algorithms collude?

Most online sellers set their prices with algorithms. These algorithms monitor one another's prices and update their own accordingly. This raises a question relevant to competition law: can algorithms that never talk to one another meet at a high price, just as competitors do when they strike a deal?

Definition 14.3

Collusion is when competitors, instead of competing, keep prices above the competitive level for their joint benefit. Reaching this outcome through the interaction of pricing algorithms, without an explicit agreement, is called **algorithmic collusion**.

The model is as follows. Two firms sell the same product. If both charge the monopoly price, the total profit is 100 and each gets 50. If one undercuts the price slightly, then as in Bertrand competition the cheaper firm takes the whole market: it earns 100 that period and the other 0. If prices eventually fall to cost, profit vanishes. The one-shot game is the Prisoner's Dilemma: whatever the other side does, undercutting is more profitable, yet the equilibrium leaves both firms with 0.

If the game is repeated every period, the situation changes. The firms can use the following strategy:

Definition 14.4

Grim trigger strategy: charge the high (monopoly) price until the rival undercuts; if the rival undercuts even once, switch to the competitive price forever.

This strategy sustains cooperation with a threat. Deviating pays today but destroys all future profits. Whether deviating is profitable depends on how much the firm values the future, that is, on the discount factor δ.

Let us compare the present values of the two paths. On the cooperation path the firm gets 50 every period. Its present value is a geometric series:

$$50 + 50\delta + 50\delta^2 + \ldots = 50 / (1 - \delta)$$

This sum is found as follows. Let the series be S. Since $\delta S = 50\delta + 50\delta^2 + \ldots$, we have $S - \delta S = 50$, which gives $S = 50/(1 - \delta)$. On the deviation path the firm gets 100 today, then the trigger strategy kicks in and it gets 0 forever. Its present value is 100. For cooperation to continue, the value of cooperating must not be smaller than that of deviating:

$$50/(1 - \delta) \geq 100 \iff 50 \geq 100(1 - \delta) \iff 1 - \delta \leq 1/2 \iff \delta \geq 1/2$$

In the second step both sides were multiplied by $(1 - \delta)$; since $\delta < 1$ this number is positive, so the direction of the inequality does not change. The result: if the firms value the future enough ($\delta \geq 1/2$), nobody wants to undercut and the high price persists. This structure is the same as in the repeated Prisoner's Dilemma.

Calvano et al. (2020) studied what this theoretical result means for algorithms with a simulation. Reinforcement-learning pricing agents (which update their pricing rule every period according to the profit signal) were able to learn prices above the competitive level without ever messaging one another. Moreover, when one agent undercut, the other responded with a temporary price war; that is, the agents spontaneously learned a punishment behavior resembling the trigger strategy.

This finding raises a regulatory question. Competition law is traditionally built on the notion of an *agreement between people*: competitors meeting to discuss prices is prohibited. Algorithms that learn a high price without communicating explicitly strain this framework. If there is no agreement, is there collusion? Who is responsible: the person who wrote the algorithm or the one who uses it? These questions rest on the results of game theory: repeated interaction and the value of the future can sustain cooperation even without an explicit agreement.

14.2.5 Code trace: compare()

To see the trigger condition in numbers, the cumulative value of the two paths can be computed period by period. The function below compares, over T periods, the path that cooperates with 50 each period against the deviation path that undercuts in period s, earns 100 in that period, and then earns 0.

```
def compare(delta, s, T):
    coop, dev = 0.0, 0.0          # cooperate / deviate
    for t in range(T):
        coop += delta**t * 50
        if t < s:
            dev += delta**t * 50
        elif t == s:
            dev += delta**t * 100  # A undercuts
        # t > s: trigger fired, profit 0
    return round(coop, 2), round(dev, 2)
print(compare(0.6, 2, 6))
print(compare(0.4, 2, 6))
```

The code runs line by line as follows:

- `coop` is the cumulative present value of the cooperation path and `dev` that of the deviation path; both start at 0.
- The loop runs over the periods `t = 0, 1, …, T − 1`. The profit of each period is multiplied by `delta**t`; this is the present value of a profit received t periods later.
- The cooperation path adds 50 every period.
- The deviation path cooperates before the moment of deviation (`t < s`) and adds 50. At the moment of deviation (`t == s`) it takes the whole market and adds 100. After that (`t > s`) the trigger is in effect and nothing is added.
- The function returns the two values rounded to two decimal places.

The first call is for $\delta = 0.6$, $s = 2$, $T = 6$. The trace is in Table 14.4.

t	δ^t	cooperation adds	coop	deviation adds	dev
0	1	50	50	50	50
1	0.6	30	80	30	80
2	0.36	18	98	36 (deviation)	116
3	0.216	10.8	108.8	0	116
4	0.1296	6.48	115.28	0	116
5	0.0778	3.89	119.17	0	116

Table 14.4: Trace of `compare(0.6, 2, 6)`.

In the first two periods the two paths are identical. At $t = 2$ the deviating firm takes the whole market: $0.36 \cdot 100 = 36$; the cooperation path adds only $0.36 \cdot 50 = 18$, so deviation moves ahead by 18. Then the trigger kicks in; the deviation path no longer adds anything, while the cooperation path keeps adding every period. At $t = 4$ the gap falls to 0.72, and at $t = 5$ cooperation pulls ahead: 119.17 > 116. The short-term gain of 18 has been paid back by the lost future profits.

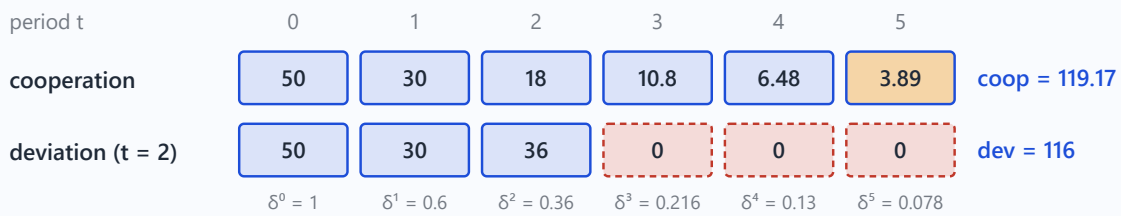


Figure 14.5: Present values of the per-period profits of the two paths for $\delta = 0.6$. Deviation yields 36 at $t = 2$, after which three periods of 0 follow (dashed red boxes). The six-period totals are 119.17 for cooperation and 116 for deviation.

Figure 14.5 shows the six-period window. In the infinite horizon the gap is even larger: the value of cooperation is $50/(1 - 0.6) = 125$, while that of deviation stays at 116 (nothing is added after the moment of deviation). So at $\delta = 0.6$ deviation is not profitable, which is consistent with the condition $\delta \geq 1/2$.

The second call is for $\delta = 0.4$. This time the future loses value faster.

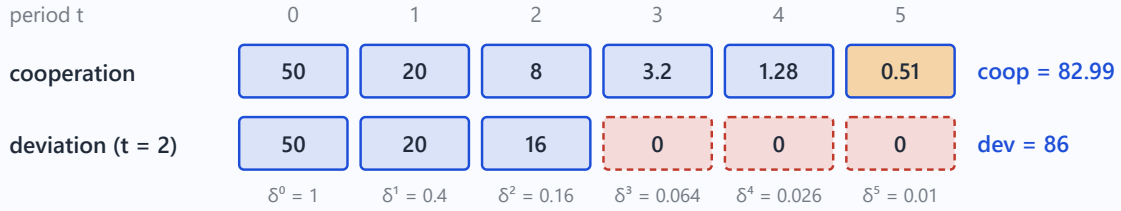


Figure 14.6: The same comparison for $\delta = 0.4$. The gain from deviating at $t = 2$ is 16, and the later losses do not make up for it: cooperation 82.99, deviation 86.

In Figure 14.6 the cooperation path adds 50, 20, 8, 3.2, 1.28, and 0.51, reaching 82.99. The deviation path adds 50, 20, and 16 and stays at 86. In the infinite horizon the value of cooperation is $50/0.6 \approx 83.3$, which is again smaller than 86. Deviation is profitable and the collusion collapses. The threshold is $\delta = 1/2$: at this value the two paths are equal in the limit ($50/0.5 = 100$).

Common mistake

Deciding from a finite window. The code sums 6 periods; this is only an illustration. The trigger condition is defined over the infinite horizon. At $\delta = 0.6$ the gap after 6 periods is 3.17, while in the limit it is $125 - 116 = 9$. If δ is very close to the threshold, a short window can give

the wrong conclusion.

14.2.6 Conditions that facilitate and hinder collusion

The trigger condition also tells us in which markets collusion is more likely. The first factor is the number of firms. If n firms share the monopoly profit equally, each gets $100/n$ per period. A deviating firm still takes the whole market (100) for one period and then 0. The condition becomes:

$$(100/n) / (1 - \delta) \geq 100 \iff 1/n \geq 1 - \delta \iff \delta \geq 1 - 1/n$$

Firms n	Share 100/n	Required δ
2	50	0.50
3	33.3	0.67
4	25	0.75
5	20	0.80
10	10	0.90

Table 14.5: The cooperation share by number of firms and the smallest required δ.

As firms multiply, the cooperation share shrinks while the payoff from deviating stays the same. Sustaining collusion in a crowded market therefore takes much more patience. The $\delta = 0.6$ that suffices with two firms is not enough in a market with five.

Example 14.3

Three firms share the monopoly profit (100) and $\delta = 0.7$. Does collusion hold?

1. Under cooperation each firm's share is $100/3 \approx 33.3$; its value is $33.3/(1 - 0.7) = 33.3/0.3 \approx 111.1$.
2. The value of deviating is 100.
3. Since $111.1 \geq 100$, deviation is not profitable and collusion holds. By the formula too: $\delta = 0.7 \geq 1 - 1/3 \approx 0.67$ ✓.
4. If a fourth firm enters, the threshold rises to 0.75. $\delta = 0.7$ is no longer enough: $25/0.3 \approx 83.3 < 100$. The entry of a new competitor breaks the collusion.

Besides the number of firms, three more factors matter. They explain why algorithmic pricing can make collusion easier.

Frequent price updates

δ depends on the length of a period. If prices change once a year, a profit one period later loses considerable value relative to today. If prices are updated every minute, a profit one minute later is worth almost as much as today's; $\delta \approx 1$ per period. Moreover, the deviator's gain is limited to the few minutes until the rival reacts. Both effects make collusion easier.

Transparent prices

If the rival's price is visible instantly, a deviation is noticed immediately and the punishment is not delayed. If a deviation could be noticed only several periods later, the deviator would earn 100 in all those periods and the threshold would rise.

Price-matching guarantees

A guarantee such as "if a competitor is cheaper, we pay the difference" looks consumer-friendly at first sight. Yet it wipes out the gain from undercutting: if the rival cuts its price, the customer still buys from the firm offering the guarantee. Since undercutting wins no customers, nobody undercuts and the collusion is preserved.

Regulatory proposals

Test a pricing algorithm against other algorithms in a simulation environment before releasing it to the market, and open the algorithm's decision logs to audit. These proposals rest on the observation that collusion can arise from a dynamic rather than from an agreement.

Common mistake

Forgetting the number of firms. The patience that suffices with two firms is not enough in a crowded market. The threshold is $\delta \geq 1 - 1/n$; for $n = 5$ it is 0.80. Memorizing the condition as always $\delta \geq 1/2$ is wrong.

14.2.7 Back to the tournament

The first game of the opening tournament is the Prisoner's Dilemma. The payoffs are (3, 3) for mutual cooperation, (0, 5) or (5, 0) between a cooperator and a defector, and (1, 1) for mutual defection. Each match is 60 rounds. Since the Random agent decides with probability one half each round, its matches are computed with expected values. The Best Response agent cooperates in the first round. After that it plays the best response to the opponent's most frequent move. In the Prisoner's Dilemma defection is the dominant strategy, so the best response is defection whatever the opponent does. That is, the Best Response agent always defects after the first round.

Example 14.4

Let us compute the expected total of each agent's three matches one by one.

1. **Always Defect.** Against Always Cooperate it gets 5 every round: $60 \cdot 5 = 300$. Against Random it gets 5 or 1 with equal probability each round, 3 in expectation: $60 \cdot 3 = 180$. Against Best Response it gets 5 in the first round (the opponent cooperates) and 1 in the next 59 rounds: $5 + 59 = 64$. The total is $300 + 180 + 64 = 544$.

2. **Best Response.** Against Always Cooperate it gets 3 in the first round, then 5 for 59 rounds: *3 + 295 = 298*. Against Always Defect it gets 0 in the first round, then 1 for 59 rounds: 59. Against Random it gets *(3 + 0)/2 = 1.5* in expectation in the first round, then *(5 + 1)/2 = 3* every round: *1.5 + 177 = 178.5*. The total is *535.5*.
3. **Random.** Against Always Cooperate it gets *(3 + 5)/2 = 4* every round: 240. Against Always Defect it gets *(0 + 1)/2 = 0.5* every round: 30. Against Best Response it gets 4 in the first round, then 0.5 for 59 rounds: *4 + 29.5 = 33.5*. The total is *303.5*.
4. **Always Cooperate.** Against Always Defect it gets 0. Against Random it gets *(3 + 0)/2 = 1.5* every round: 90. Against Best Response it gets 3 in the first round, then 0: 3. The total is *93*.

Agent	Expected total
Always Defect	544
Best Response	535.5
Random	303.5
Always Cooperate	93

Table 14.6: Expected totals in the Prisoner's Dilemma (Level 1, 60 rounds per match).

The answer to the first question is Always Defect. Best Response finishes a very close second; the difference between the two comes only from the cooperation in the first round. In this tournament mutual cooperation is almost never seen, because no agent rewards or punishes based on the past. Since there is no threat to sustain cooperation, the dominant strategy wins.

For the second question the game is changed. In **Matching Pennies**, one player wins if the two choices match and the other wins if they differ; the game is zero-sum and has no pure equilibrium. Here the ranking changes. Two fixed agents that always make the same move are easily exploited: the Best Response agent learns their move and wins every round after the first. This puts it first with about +118. The Random agent, on the other hand, cannot be exploited by any opponent; its expected payoff in every match is 0. This is the security value of the mixed equilibrium of Matching Pennies (playing half and half).

Real tournaments teach the same lesson. Axelrod's (1980) repeated Prisoner's Dilemma tournament was won by **Tit-for-Tat** (the strategy that cooperates in the first round and then repeats the opponent's last move). Reciprocity and the threat of punishment sustained cooperation. Among the four agents here, none punishes based on the past; so cooperation does not arise. The pricing agents in Calvano et al. (2020), by contrast, remember the previous period, and can learn to punish and sustain a high price. The difference lies in memory.

The general lesson of the tournament is this: **there is no universally best strategy**. The success of a strategy depends on the game being played, the population of opponents, noise, and memory. It is wrong to test an agent in a single environment and assume it will do well everywhere. This observation also explains why reproducible experiments across several games and several opponents are necessary to evaluate agents (14.2.8).

14.2.8 A reproducible experimental protocol

The moves of language-model agents involve randomness: the same pairing can give different scores twice. A result is reliable if someone else can repeat the same experiment and obtain the same result. This is called **reproducibility**. In an agent tournament experiment, the following are recorded:

- a fixed random seed;
- the full text of the prompt;
- the version of the model used and the temperature value;
- the order of opponents and the number of rounds: these are held fixed and stated in the report.

The report then gives the following:

- each pairing is repeated at least N times (for example $N = 30$);
- the spread together with the mean: the standard deviation or a 95% confidence interval;
- the code and the game logs are shared in an open repository.

A **confidence interval**, instead of a single mean, gives a range in which the true mean most likely lies. If the mean of N repetitions is m and the standard deviation is s, the approximate 95% confidence interval is $m \pm 1.96 \cdot s / \sqrt{N}$. Here $s/\sqrt{N}$ is the uncertainty of the mean itself; it shrinks as the number of repetitions grows.

Example 14.5

Over 30 repetitions the mean score is 2.6 and the standard deviation is 0.4.

1. $\sqrt{30} \approx 5.48$, $0.4 / 5.48 \approx 0.073$.
2. Half-width: $1.96 \cdot 0.073 \approx 0.14$.
3. Interval: 2.6 ± 0.14, that is, [2.46, 2.74].

If another agent's mean falls inside this interval, keep in mind that the difference between the two agents may be due to noise.

14.3 Learning mechanisms

In mechanism design the designer chooses the rule so as to produce the desired outcome. In the classical approach this choice is made with a proof. This section examines a second approach: learning the rule as the solution of an optimization problem or directly from data. Two new problems that learning brings are also addressed: overstating one's own success on little data, and the privacy of individuals' data.

14.3.1 Automated mechanism design

The classical way of doing mechanism design is as follows. A human proposes a rule: a second-price (Vickrey) auction, the VCG mechanism, or Myerson's auction that maximizes expected revenue. Then it is proved by hand that the rule is incentive compatible and optimal for a given objective. This approach is very successful for well-structured problems such as selling a single item. But with many items and many buyers, the mechanism that maximizes expected revenue is often unknown; there is not even a candidate to prove things about by hand.

In the automated approach the mechanism is **the solution of an optimization problem**. The problem has an objective (revenue or social welfare) and two kinds of constraints.

Definition 14.5

Incentive compatibility (IC): For every player, reporting the true type (value) is no worse than reporting any other type. In short, telling the truth is best.

Individual rationality (IR): For every player, participating in the mechanism is no worse than not participating. In short, participation does not hurt: the expected utility does not fall below zero.

Automated design takes two basic forms.

- **Linear programming.** If the players' types come from a finite set, then for each type the allocation probability and the payment are variables. The objective and the constraints are linear in these variables. Because the allocation probabilities are variables, randomized mechanisms are allowed and the problem is solved by linear programming (Conitzer and Sandholm, 2002); if only deterministic mechanisms are wanted, the problem becomes hard in general. The example in 14.3.2 is of this kind.
- **Auctions as neural networks.** If types are continuous and multidimensional, the mechanism is represented by a neural network: the network takes the bids and outputs the allocation and the payments. The IC constraint cannot be written directly; instead a **regret** penalty is added (Dütting et al., 2019). Here regret is the largest extra utility a bidder could gain by lying. Training tries to increase revenue while driving regret toward zero.

The automated approach has three limits:

Approximate IC
: The learned rule may not be exactly incentive compatible; a small regret remains after training. This is called **approximate incentive compatibility**. If the gain from lying is so small that many participants would not find it worth computing, it may not be a problem in practice, but it is not a proven guarantee.

Generalization
: The rule is learned from the value distribution in the training data. If the true distribution changes (for example, through an economic shock), the rule may become ineffective.

Explainability
: When a participant asks "Why did I pay this price?", the weights of a neural network give no understandable answer. The Vickrey rule, on the other hand, can be explained in one sentence: "the winner pays the second-highest bid".

14.3.2 Code trace: revenue()

The smallest example of automated design involves a single buyer and a single item. The buyer has two possible types: type L values the item at 1 and has probability q_L; type H values it at 3 and has probability $1 - q_L$. The seller does not know the buyer's type and asks the buyer to report it. For every reported type the mechanism specifies an allocation probability and a payment. Type H always gets the item. Type L gets it with probability x_L. The payments are p_L and p_H.

When revenue is maximized, two constraints hold with equality (they bind). They are derived as follows.

1. **L's participation constraint (IR).** The utility of type L must satisfy $1 \cdot x_L - p_L \geq 0$. To raise revenue, the seller pushes p_L up to this limit: $p_L = x_L$. The price for L is the highest amount L is willing to pay.
2. **H's incentive compatibility constraint (IC).** If type H says "I am L", it gets the item with probability x_L and pays p_L: its utility is $3x_L - p_L$. If it tells the truth, its utility is $3 - p_H$. Telling the truth must be at least as good as lying: $3 - p_H \geq 3x_L - p_L$. The seller pushes p_H up to this limit: $p_H = p_L + 3(1 - x_L)$.
3. **Expected revenue.** $G = q_L\, p_L + (1 - q_L)\, p_H$.

Once the payments are substituted, the revenue depends only on x_L:

$$G(x_L) = q_L\, x_L + (1 - q_L)(3 - 2x_L) = 3(1 - q_L) + x_L(3q_L - 2)$$

This expression is linear in x_L. Selling to L has a cost: selling cheaply to L also opens a lying option for H and lowers H's price. The code below computes the revenue for three values of x_L and two values of q_L.

```
def revenue(xL, qL, vL=1, vH=3):
    pL = xL * vL                 # L's IR binds
    pH = pL + vH * (1 - xL)      # H's IC binds
    return qL * pL + (1 - qL) * pH
for qL in [0.5, 0.75]:
    G = [revenue(x, qL) for x in [0, 0.5, 1]]
    print(qL, G)
```

The code runs line by line as follows. `pL` is the payment that makes L's participation constraint bind. `pH` is the payment that makes H's incentive compatibility constraint bind. The function returns the average weighted by the probabilities of the two types. The loop tries $x_L \in \{0, 0.5, 1\}$ for $q_L = 0.5$ and 0.75.

q_L	x_L	p_L	p_H	Revenue
0.5	0	0	3	$0.5 \cdot 3 = 1.5$
0.5	0.5	0.5	2	$0.25 + 1 = 1.25$
0.5	1	1	1	1
0.75	0	0	3	$0.25 \cdot 3 = 0.75$
0.75	0.5	0.5	2	$0.375 + 0.5 = 0.875$
0.75	1	1	1	1

Table 14.7: Trace of `revenue()`.

The row $x_L = 0.5$ shows the trade-off clearly. Selling to L with probability one half and receiving an expected payment of 0.5 (the price is 1 if a sale occurs) opens H the option "if I say L, I earn $3 \cdot 0.5 - 0.5 = 1$". To make H willing to tell the truth, H's price must fall from 3 to 2. At $q_L = 0.5$ this loss is greater than the extra revenue from L.

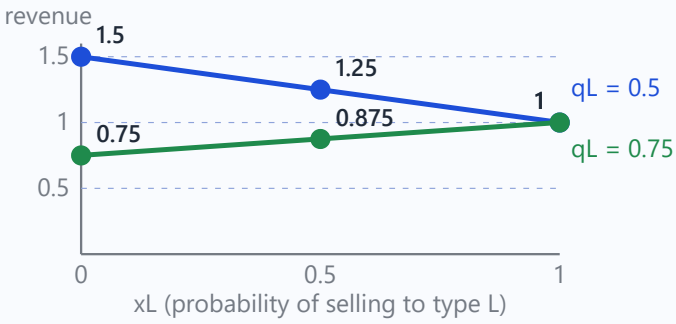


Figure 14.7: How revenue varies with x_L for the two-type buyer. At $q_L = 0.5$ (blue) revenue decreases and the best choice is $x_L = 0$; at $q_L = 0.75$ (green) revenue increases and the best choice is $x_L = 1$.

Two lines are visible in Figure 14.7. At $q_L = 0.5$ the best choice is $x_L = 0$: sell only to H at 3, for revenue 1.5. At $q_L = 0.75$ type L is the majority; selling only to H now earns little (0.75), and the best choice is to sell to everyone at 1, for revenue 1. The coefficient of x_L in the revenue formula is $3q_L - 2$. It is −0.5

at $q_L = 0.5$ and +0.25 at $q_L = 0.75$.

There is an important observation: since revenue is linear in x_L, at least one optimum is always at an endpoint (at $q_L = 2/3$ the coefficient is 0 and every x_L gives the same revenue). This is a general property of linear programming: if the feasible region is a bounded polyhedron, at least one optimum is at a vertex. Both solutions are in fact **a posted price**: either "price 3" or "price 1". So in this small example automated design rediscovers the result that Myerson's theory gives for a single item: the mechanism that maximizes expected revenue is a well-chosen reserve price. The real value of automated design emerges in multi-item settings where no such theory exists.

14.3.3 Learning a reserve price from data

In the previous example the seller knew the type distribution. In reality the distribution is unknown; all we have are values observed in the past. In that case a natural approach is to choose the price that would have brought the most revenue on the historical data.

Definition 14.6

Let n sample values $v_1, \ldots, v_n$ be given. At price p, the buyers whose value is at least p buy. The **empirical revenue** is the average revenue per buyer:

$$g(p) = p \cdot (\text{number of samples with value} \geq p) / n$$

Empirical revenue maximization is choosing the price that maximizes this function.

Which prices need to be tried is found as follows. If a price is chosen between two consecutive sample values, it can be raised without changing the number of buyers, and this increases revenue. So the best price is always one of the sample values. Sort the values from largest to smallest. If the k-th largest value is the price, exactly k buyers buy (if there are no ties). So trying only n candidate prices suffices.

```
def best_price(samples):
  v = sorted(samples, reverse=True)
  n = len(v)
  price, revenue = 0, 0
  for k in range(1, n + 1):
    p = v[k - 1]        # k-th highest
    g = p * k / n       # k buyers buy
    if g > revenue:
      price, revenue = p, g
  return price, revenue
```

The code runs line by line as follows. `v` is the samples sorted from largest to smallest. `price` and `revenue` hold the best price so far and its empirical revenue. For $k = 1, \ldots, n$ the loop tries the k-th largest value as the price. If the values are all distinct, k buyers buy at this price and the empirical revenue is $p \cdot k / n$. (If several samples share the same value, all of them buy; the last position of that value gives the correct count, and the largest revenue is still found correctly.) If the revenue exceeds the previous best, the best price is updated.

In the example, the 8 buyer values observed in the past are 0.62, 0.15, 0.91, 0.47, 0.33, 0.78, 0.55, 0.08. Sorted, they are 0.91, 0.78, 0.62, 0.55, 0.47, 0.33, 0.15, 0.08.

k	p	g = p·k/8	Best (price, revenue)
1	0.91	0.114	(0.91, 0.114) new
2	0.78	0.195	(0.78, 0.195) new
3	0.62	0.2325	(0.62, 0.2325) new
4	0.55	0.275	(0.55, 0.275) new
5	0.47	0.294	(0.47, 0.294) new
6	0.33	0.2475	unchanged
7	0.15	0.131	unchanged
8	0.08	0.08	unchanged

Table 14.8: Trace of `best_price()`, $n = 8$.

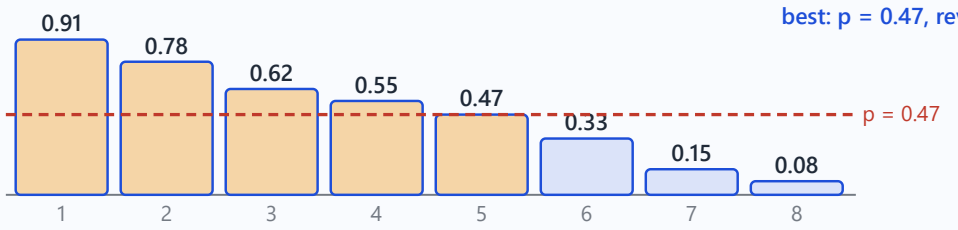


Figure 14.8: The 8 sorted sample values and the learned price. At $p = 0.47$, the 5 buyers whose value is at least 0.47 (orange) buy; the empirical revenue is $0.47 \cdot 5/8 \approx 0.294$.

Figure 14.8 shows the learned price. As the price falls, the number of buyers rises but the money taken from each buyer shrinks. In the first five steps the effect of more buyers dominates; at the sixth step (0.33) the effect of the price drop starts to dominate. The learned price is 0.47 and the empirical revenue is about 0.294.

14.3.4 Empirical versus true revenue

What the learned price will actually earn in the future depends on the true distribution from which the values come. For example, suppose the values come from the distribution $U[0, 1]$. This means that intervals of equal length between 0 and 1 are equally likely. In this distribution the probability that a

buyer's value exceeds p is $1 - p$. So the **true revenue** per buyer at price p is:

$$G(p) = p \cdot (1 - p)$$

This function is a parabola. Its maximum is at the midpoint: at $p = 0.5$, $G = 0.5 \cdot 0.5 = 0.25$. The true revenue of the learned price is $0.47 \cdot 0.53 \approx 0.249$. So the learned price is almost as good as the best one. But the empirical revenue (0.294) is clearly higher than the true revenue (0.249).

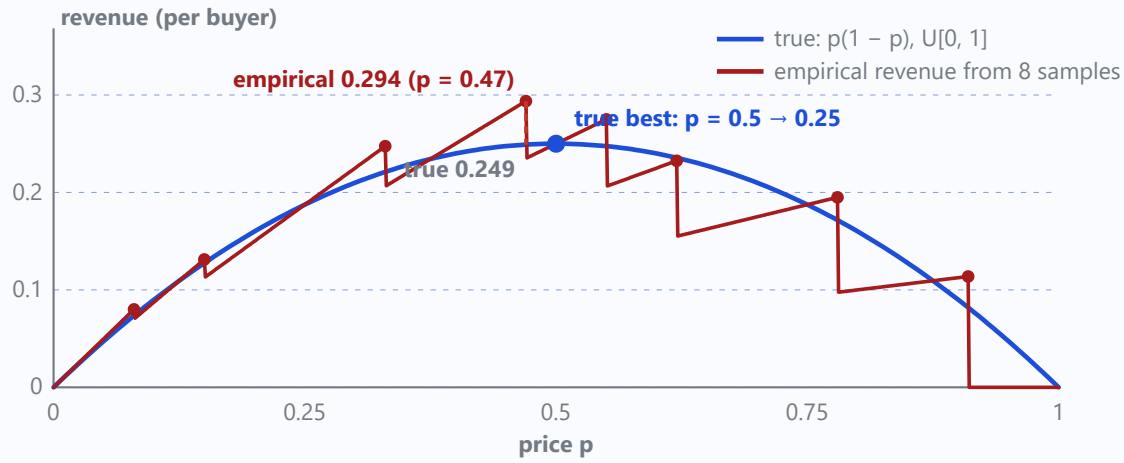


Figure 14.9: The empirical revenue curve (red, piecewise linear: it jumps down at the sample values) and the true revenue curve $p(1 - p)$ (blue). The empirical curve jumps at the sample values and its highest point is at a sample value (0.47). The true best is 0.25 at $p = 0.5$.

Figure 14.9 compares the two curves. The empirical curve is like a sawtooth: between two sample values it rises linearly with the price, and when it passes a sample value it drops because that buyer is lost. The true curve is a smooth parabola. The empirical curve lies above the truth at some prices and below it at others.

That the empirical success of the chosen price is overstated is no coincidence. The price was chosen as the *highest* point of the empirical curve. The places where the empirical curve rises above the truth are places where the samples happen to be plentiful by chance. When the highest point is chosen, one of these lucky fluctuations is chosen too. Because the data that is measured is the same as the data that is selected, an **optimistic bias** arises. The gap between the true best revenue (0.25) and the true revenue of the chosen price (0.249) is small; but as the sample shrinks, both this gap and the optimistic bias grow.

Common mistake

Taking empirical revenue for the truth. The success of a price that is selected and measured on the same data is overstated. In this example 0.294 is measured but the truth is 0.249; the gap is 18%. To measure the success of a price, an auction rule, or any learned model, separate

data that was not used in the selection is needed.

14.3.5 How many samples are enough? Sample complexity

As the number of samples grows, the empirical curve approaches the truth. A simulation shows this: draw n values from $U[0, 1]$, maximize the empirical revenue, and record the empirical and true revenue of the chosen price. This is repeated hundreds of times for each n and averaged.

Samples n	Empirical revenue	True revenue	0.25 – true
10	0.311	0.224	0.0262
100	0.263	0.245	0.005
1.000	0.253	0.249	0.0011
10.000	0.25	0.25	0.0002

Table 14.9: Empirical and true revenue by number of samples (values $U[0, 1]$, average of hundreds of repetitions). The best revenue is 0.25 ($p = 0.5$).

The table shows two things. First, **overfitting**: with few samples the empirical revenue looks much higher than the true one (0.311 against 0.224 at $n = 10$). The learned price fits the features of those particular samples rather than of the true distribution; in a sense it has "memorized" the samples. Second, as the samples grow, the two columns approach each other and 0.25. The loss in true revenue falls to roughly one fifth with every tenfold increase in samples.

Definition 14.7

Sample complexity: The minimum number of samples needed to find, with high probability (for example 95%), a price that is as close to the best revenue as desired (for example at most ε short of it).

Cole and Roughgarden (2014) studied this question for auctions. Roughly, the result is this: if the value distribution does not produce extreme values, the number of samples needed for a good mechanism grows **moderately** (polynomially) in the number of bidders and the desired accuracy. So the sample count does not explode for "a bit more accuracy". For distributions with extreme values, no such guarantee may be possible. The reason is intuitive: if very rare but very large values make up a significant part of the revenue, a great many samples are needed to see these values often enough.

The lesson of this subsection is familiar from machine learning: a mechanism learned from data overstates its own success on little data. Separating training and evaluation data is as necessary in mechanism design as in machine learning.

14.3.6 Separate data for choosing and measuring

The simplest way to prevent optimistic bias is to split the data in two. The **training data** is used to choose the price and the **validation data** to measure the chosen price. Because the validation data was not used in the selection, a measurement made on it is not biased.

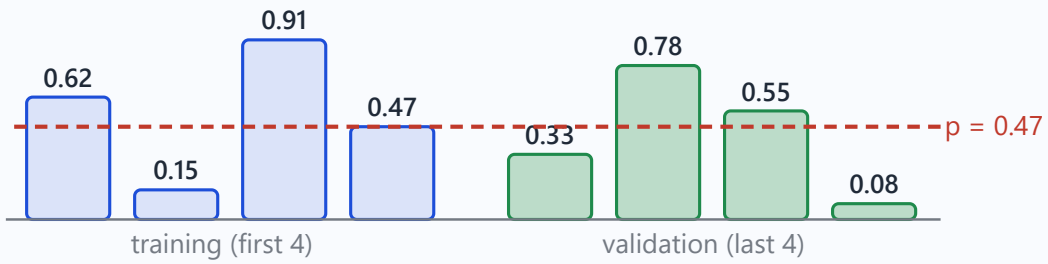


Figure 14.10: Of the 8 samples, the first 4 are set aside for training and the last 4 for validation. The price chosen on the training data is 0.47; in the validation data only 0.78 and 0.55 lie above this price.

Example 14.6

Use the split in Figure 14.10: training = 0.62, 0.15, 0.91, 0.47 and validation = 0.33, 0.78, 0.55, 0.08.

1. **Choosing the price on the training data.** The sorted training values are 0.91, 0.62, 0.47, 0.15. The empirical revenues are $0.91 \cdot 1/4 \approx 0.228$, $0.62 \cdot 2/4 = 0.31$, $0.47 \cdot 3/4 \approx 0.353$, $0.15 \cdot 4/4 = 0.15$. The best is $p = 0.47$, with training revenue 0.353.
2. **Measuring on the validation data.** Of the validation values, 0.78 and 0.55 are above the price: $0.47 \cdot 2/4 = 0.235$.
3. **Comparison.** The true revenue is $0.47 \cdot 0.53 \approx 0.249$. The training measurement (0.353) is very optimistic; the validation measurement (0.235) is close to the truth.
4. **When the roles are swapped.** If training = the last 4, the sorted values are 0.78, 0.55, 0.33, 0.08 and the revenues are 0.195, 0.275, 0.2475, 0.08. The chosen price is 0.55. In the first 4 there are 0.62 and 0.91 above this price: $0.55 \cdot 2/4 = 0.275$.
5. **Cross-validation.** The two validation estimates are averaged: $(0.235 + 0.275)/2 = 0.255$.

Measurement	Calculation	Revenue
On the training data (where it was chosen)	$0.47 \cdot 3/4$	0.353
On the validation data	$0.47 \cdot 2/4$	0.235
True, $U[0, 1]$	$0.47 \cdot 0.53$	0.249

Table 14.10: Three different measurements of the price $p = 0.47$.

This method has a cost: the price is now chosen with only 4 samples rather than 8, so it is noisier. Cross-validation reduces this cost by using all the data in turn for both selection and measurement.

There is one more risk specific to mechanisms. In machine learning the data usually comes from nature; the data's producers do not care what the model will do. In an auction, however, the data comes from the bidders themselves. If buyers know that their past bids determine the future price, they can deliberately bid low today: they lose a little but face a lower reserve price in the future. That is, **the data can also be strategic**. In that case the learning algorithm is itself a mechanism, and it too must be incentive compatible.

14.3.7 Differential privacy and the exponential mechanism

Another problem with learning from data is privacy. Once the learned price is announced, it leaks something about the values of past buyers. For example, if adding the data of a single buyer changes the price a great deal, that buyer's value can be inferred from the price.

Definition 14.8

Let M be a mechanism with random output. For every pair of datasets D and D' that differ in the data of only one person, and for every outcome o,

$$P[M(D) = o] \le e^{\varepsilon} \cdot P[M(D') = o]$$

if this holds, then M is **ε-differentially private**.

In plain terms: even if one person's data is added or removed, the probability of every outcome changes only slightly. ε is the measure of privacy. A small ε means more privacy: the probabilities can change by a factor of at most e^{ε}. $\varepsilon = 0$ is perfect privacy, but then the output cannot depend on the data at all; as ε grows the output tracks the data more closely but privacy decreases. The definition requires a randomized mechanism. A deterministic rule can jump from one outcome to another with one person's data; the probability of that outcome drops from 1 to 0 and the ratio becomes infinite.

McSherry and Talwar (2007) gave a general tool that satisfies this definition.

Definition 14.9

Exponential mechanism: Each candidate outcome p has a score $R(p)$ (for example the revenue of the price). Δ is the most a score can change when a single person is removed. The mechanism picks each candidate with probability proportional to its weight:

$$P[p \text{ is chosen}] = \exp(\varepsilon \cdot R(p) / 2\Delta) / \Sigma_{p'} \exp(\varepsilon \cdot R(p') / 2\Delta)$$

The mechanism does not choose the best price with certainty; it draws a lottery. Good prices get exponentially higher probability; bad prices get a small but nonzero probability. The reason for the 2 in the formula is this: when one person is removed, both the numerator (the weight of the chosen candidate) and the denominator (the sum of all weights) can change. Each changes by a factor of at most $e^{\varepsilon/2}$; together they produce a change of at most a factor of e^{ε}. This is the bound in the definition.

This mechanism has a strong connection with game theory. Because a single person's report affects the outcome little, the gain that person can obtain by lying is also small. If utilities lie, for example, in the interval [0, 1] and the person changes their own report, then under ε-differential privacy the extra gain from lying is at most $e^{\varepsilon} - 1$. So differential privacy yields **approximate incentive compatibility**, and does so together with privacy. This is a surprising result: privacy and incentives can be handled with the same mathematical tool.

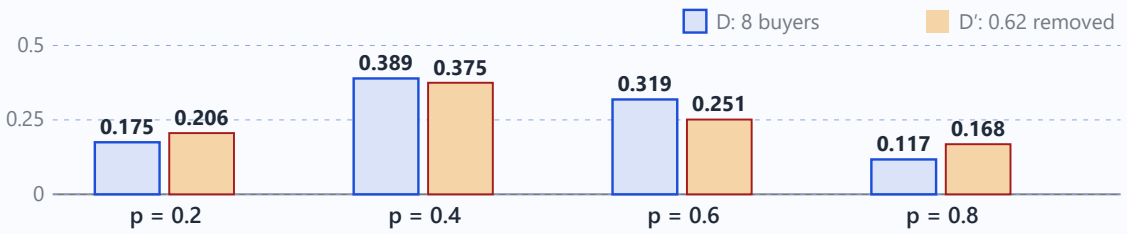


Figure 14.11: Selection probabilities of the exponential mechanism in the reserve price example, $\varepsilon = 2$. Blue: the data of the 8 buyers (D); orange: with the buyer of value 0.62 removed (D′). The two distributions are close to each other; the largest ratio is 1.44.

Figure 14.11 visualizes what the definition says. When one buyer is removed from the data, the selection probabilities change, but only a little. In this example the largest change is by a factor of 1.44; the bound the definition permits is $e^2 \approx 7.39$. The step-by-step calculation of the probabilities is in 14.3.8.

Common mistake

Assuming a linear proportion in the exponential mechanism. The probability is proportional not to the revenue but to $\exp(\varepsilon \cdot R / 2\Delta)$. A revenue difference translates exponentially into the probability ratio: if two candidates differ by 1 in score and $\varepsilon = 2$, $\Delta = 1$, the probability ratio is $e^1 \approx 2.72$, not the ratio of the scores.

14.3.8 Code trace: exp_mech()

The code below applies the exponential mechanism with the 8 buyer values from the example of learning a price from data. The candidate reserve prices are 0.2, 0.4, 0.6, 0.8. The score is the price times the number of buyers who would buy at that price (the total revenue). Since the prices are

below 1, removing one buyer changes the score of a price by at most 1; hence $\Delta = 1$. With $\varepsilon = 2$ the weight formula simplifies to $exp(2R/2) = e^R$.

```
import math
def exp_mech(v, prices, eps):
    R = [p * sum(x >= p for x in v) for p in prices]
    w = [math.exp(eps * r / 2) for r in R]   # Δ = 1
    W = sum(w)
    return [x / W for x in w]
v = [0.62, 0.15, 0.91, 0.47, 0.33, 0.78, 0.55, 0.08]
P = [0.2, 0.4, 0.6, 0.8]
q = exp_mech(v, P, 2)
print([round(x, 3) for x in q])
q = exp_mech(v[1:], P, 2)       # 0.62 removed
print([round(x, 3) for x in q])
```

The code runs line by line as follows. `R` computes the score of each candidate price; `sum(x >= p for x in v)` counts the buyers whose value is at least the price. `w` holds the weights. `W` is the sum of the weights. The function divides each weight by the sum and returns the probabilities. The code is run first with all the data and then with the first buyer (0.62) removed.

p	R (D)	$w = e^R$	q (D)	R (D′)	w′	q (D′)	Ratio
0.2	*0.2 · 6 = 1.20*	3.32	0.175	*0.2 · 5 = 1.00*	2.72	0.206	1.18
0.4	*0.4 · 5 = 2.00*	7.39	0.389	*0.4 · 4 = 1.60*	4.95	0.375	1.04
0.6	*0.6 · 3 = 1.80*	6.05	0.319	*0.6 · 2 = 1.20*	3.32	0.251	1.27
0.8	*0.8 · 1 = 0.80*	2.23	0.117	*0.8 · 1 = 0.80*	2.23	0.168	1.44
Total		W = 18.98	1		W′ = 13.22	1	

Table 14.11: Trace of `exp_mech()`, $\varepsilon = 2, \Delta = 1$. D: 8 buyers; D′: 0.62 removed.

With all the data the most likely price is $p = 0.4$ (38.9%); but it is not alone, and even the bad price 0.8 has an 11.7% chance. The differences among the weights are exponential: when the score rises from 0.8 to 2, the weight rises from 2.23 to 7.39. When the buyer of value 0.62 is removed, the scores of $p = 0.2$, 0.4, and 0.6 fall; the score of $p = 0.6$ falls the most (by 0.6). The score of $p = 0.8$ does not change, because 0.62 was already below 0.8. Since the total weight shrinks, the probability of $p = 0.8$ increases.

The last column is the ratio of the two distributions (the larger over the smaller). The largest ratio is 1.44, far below the bound $e^2 \approx 7.39$. The data of a single person affects the outcome only to a bounded extent; the mechanism is ε-differentially private. The bound is for the worst case; on a typical dataset the actual change is usually much smaller.

14.4 Blockchain mechanisms

Blockchains are systems that enforce rules with public code. This section examines them not as a technology but as a setting for mechanism design. Three mechanisms are covered: transaction fee auctions that allocate scarce block space, the automated market maker that sets prices with a formula, and the value arising from the power to determine the order of transactions (MEV). The concrete rules and tools used in this area change often; the notes emphasize the lasting principles.

14.4.1 Blockchains and smart contracts

Definition 14.10

Smart contract: A program whose code is public and which runs automatically on a blockchain when specified conditions occur. From the mechanism design point of view, it is a self-enforcing mechanism.

The importance of this definition for game theory is as follows. Mechanism design assumes that the designer will stick to the announced rules afterwards, that is, that there is a **commitment**. For example, in a second-price auction the seller cannot say, after seeing the bids, "actually you will pay the first price". In real life this assumption rests on trust or on law. In a smart contract, commitment is provided by code: once the rule is published, nobody can change it.

But this setting also brings a new problem. Pending transactions are visible to everyone before they enter a block. Seeing in advance what a transaction will do opens a way to profit from it. There is also a **block producer** that places transactions in blocks. The producer decides which transactions enter the block and in what order. Block space is limited; that is, it is a scarce resource, and a scarce resource is naturally allocated by an auction.

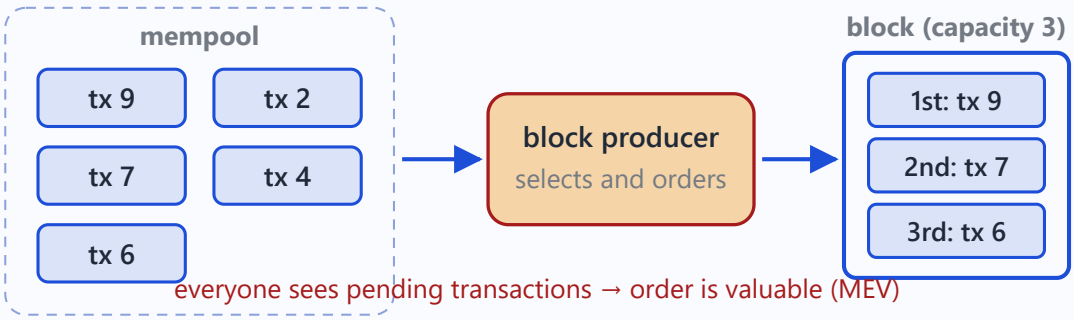


Figure 14.12: Transactions in the waiting pool go to the block producer; the producer selects and orders as many transactions as fit the capacity. Because everyone can see the pending transactions, the order is valuable.

Figure 14.12 shows the process. Users send their transactions to the **mempool**. The block producer selects transactions from the pool and orders them. Three mechanism design questions arise in this structure:

- **Transaction fee auction:** How should limited block space be allocated among users who bid transaction fees?
- **Decentralized exchange:** Can the counterparty of a trade be a formula instead of a person? This is the idea of the **automated market maker** (AMM).
- **On-chain governance:** Token-weighted voting is used to change the rules of a system. The known problems of voting rules (strategic voting, Condorcet cycles, the power of large holders) apply here too.

14.4.2 Transaction fee auctions

The number of transactions that fit in a block is limited. If more transactions are waiting, a rule decides which ones enter. This rule is called a **transaction fee mechanism**. Let us compare two designs.

The first-price design. In the design used by the first blockchains, each transaction bids a fee. The producer takes the highest bids and everyone pays their own bid. This is the familiar first-price auction. Suppose five transactions are pending with values 9, 7, 6, 4, 2, and the block capacity is 3. If everyone bids honestly, the three highest transactions enter and the producer is paid $9 + 7 + 6 = 22$.

But in a first-price auction, honest bidding is not the best strategy. A user whose value is 9 gets no surplus by bidding 9; the user wants to bid the smallest amount that is enough to get in. So everyone shades their bid. The amount of shading depends on a guess about what the others will bid. As a result, every user struggles with the question "how much should I bid?" This is tiring, and users who cannot make the guess either overpay or are left out.

The base fee + tip design. In the design known in the literature as EIP-1559 (Ethereum Improvement Proposal), the protocol sets a **base fee** r for every block. Every transaction that is willing to pay the base fee enters. The base fee is **burned**: it is paid to no one and leaves circulation. Only a small **tip** added by the user goes to the producer.

Example 14.7

The same five transactions (values 9, 7, 6, 4, 2), capacity 3, base fee $r = 5$, tip 1.

1. For a transaction to enter, its value must be at least $r + tip = 6$. The transactions 9, 7, and 6 clear this threshold; 4 and 2 do not.
2. 3 transactions enter; this does not exceed capacity, so no auction is needed.
3. Burned: $3 \cdot 5 = 15$. Tips going to the producer: $3 \cdot 1 = 3$.
4. The users' net utilities: $9 - 6 = 3$, $7 - 6 = 1$, $6 - 6 = 0$.

No guesswork is needed on the user side: each user only answers the question "is my value greater than 6?"

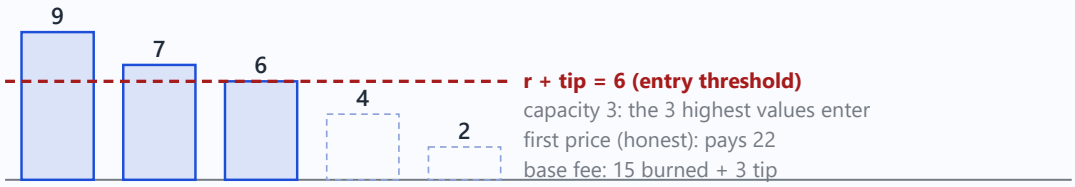


Figure 14.13: The values of the five transactions and the entry threshold $r + tip = 6$. The three transactions above the threshold enter. Under first price, 22 is paid with honest bids; in the base fee design 15 is burned and 3 goes to the producer as tips.

Figure 14.13 compares the two designs on the same data. The main advantage of the base fee design is simplicity: if demand does not exceed capacity, the best strategy is to pay "base fee + small tip", and no price guessing is needed. This resembles buying at a posted price. When demand exceeds capacity, however, the design reverts to an auction: users compete on the tip.

Roughgarden (2021) formally defined the goals of a good transaction fee mechanism: simple bidding for the user, honest behavior for the producer, and resistance to collusion. It is hard to find a rule that satisfies all of these goals completely. These goals are treated in detail in 14.4.4.

14.4.3 The base fee adjusts to demand

The base fee is not fixed; it tracks demand. The general principle is simple: if a block is above the target fullness the fee rises, and if it is below the target the fee falls. With the parameters of the original design, the update rule is:

$$r_{t+1} = r_t \cdot (1 + \tfrac{1}{8} \cdot (g_t - g_{target}) / g_{target})$$

Here g_t is the space used in block t, and g_{target} is half the block capacity. The parts of the formula read as follows. $(g_t - g_{target})/g_{target}$ is the relative deviation of fullness from the target. If the block is completely full, $g_t = 2g_{target}$ and the deviation is +1. If the block is empty the deviation is −1. At the target the deviation is 0. Because the deviation is multiplied by ⅛, the fee rises or falls by at most 12.5% in a block.

Example 14.8

Initially $r = 100$. The fullness levels of eight blocks are 100%, 100%, 100%, 100%, 50%, 25%, 0%, 0%.

1. Block 1 is 100% full: deviation +1, change +12.5%. $r = 100 \cdot 1.125 = 112.5$.

2. Block 2 is 100% full: $r = 112.5 \cdot 1.125 \approx 126.6$.
3. Block 3 is 100% full: $r \approx 126.6 \cdot 1.125 \approx 142.4$.
4. Block 4 is 100% full: $r \approx 160.2$. In a demand surge the fee rises by at most 12.5% per block.
5. Block 5 is 50% full: deviation 0, the fee stays the same: $r \approx 160.2$.
6. Block 6 is 25% full: deviation $(0.25 - 0.5)/0.5 = -0.5$, change $-0.5/8 = -6.25\%$. $r \approx 160.2 \cdot 0.9375 \approx 150.2$.
7. Block 7 is 0% full: deviation −1, change −12.5%. $r \approx 150.2 \cdot 0.875 \approx 131.4$.
8. Block 8 is 0% full: $r \approx 131.4 \cdot 0.875 \approx 115$. Empty blocks lower the fee; when demand falls, the price comes back.

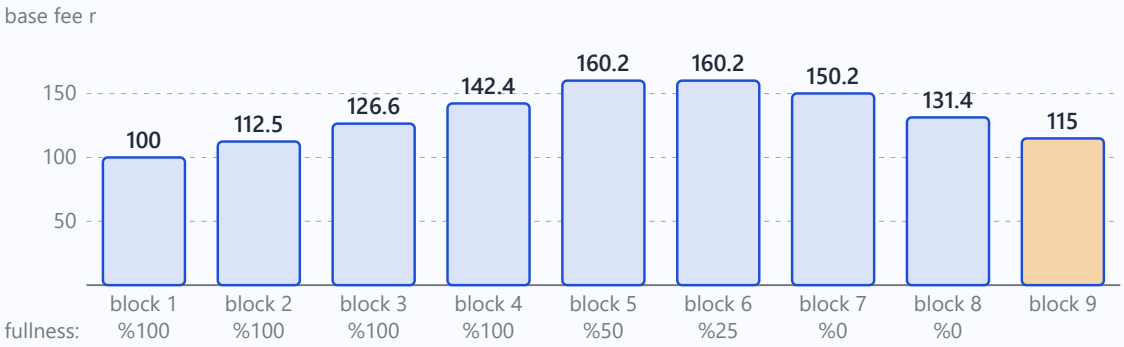


Figure 14.14: The base fee at the start of each block and the fullness of that block. Four full blocks raise the fee from 100 to 160.2; the block at the target leaves it unchanged; the partly full and empty blocks lower it to 115.

Figure 14.14 shows how the fee tracks demand. The base fee works like a **posted price** that follows demand. When demand rises, the price rises too and blocks return to the target fullness. But adjustment takes time: in a sudden demand surge the fee can rise only in steps of 12.5%. Meanwhile the blocks are full and the system temporarily reverts to a tip-based auction.

The change is **multiplicative**. After four full blocks the fee is $100 \cdot 1.125^4 \approx 160.2$, not $100 + 4 \cdot 12.5 = 150$. Moreover, an increase is not exactly undone by a decrease of the same percentage: since $1.125 \cdot 0.875 \approx 0.984$, one full and one empty block lower the fee by 1.6%.

Common mistake

Updating the base fee additively. The change is multiplicative: two full blocks give $100 \to 112.5 \to 126.6$, not 125. A second mistake is to think that the base fee goes to the producer. The base fee is burned; only the tip goes to the producer.

14.4.4 Why burn it?

Burning the base fee instead of giving it to the producer looks wasteful at first sight. The reason for this choice is explained by three properties of a good fee mechanism.

User incentive compatibility
: If demand does not exceed capacity, the best bid for a user is the base fee plus a small tip. The user does not wonder "how much should I shade my bid?"

Producer incentive compatibility
: The producer earns the most by following the rule. Adding fake transactions or hiding real ones does not pay.

Collusion resistance
: The producer and some users cannot make a deal off-chain to increase their joint gain. For example, a side deal such as "pay the tip off-chain and keep the base fee low" should not pay.

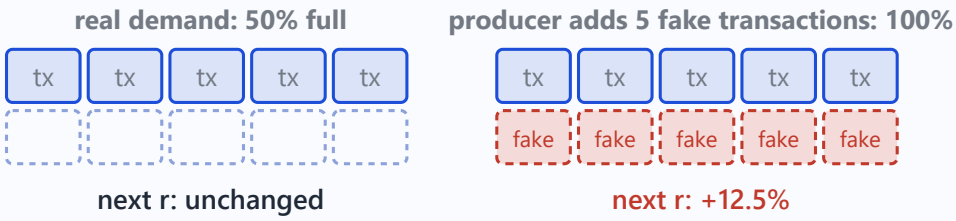


Figure 14.15: If real demand fills the blocks to 50%, the fee does not change. If the producer adds 5 fake transactions and fills the block to 100%, the next base fee rises by 12.5%.

The rationale for burning is seen in Figure 14.15. If the base fee went to the producer, it would be free for the producer to add its own fake transactions to the block: the fee it paid would return to itself. These fake transactions would fill the block and cause the base fee to rise in the next block. The producer could thus inflate the fee as it wished. When the fee is burned, every fake transaction is a real loss of r: 5 fake transactions burn $5r$. Creating fake demand is now expensive.

Burning does not solve every problem. When blocks are full, users still compete on the tip and first-price behavior returns. This reflects a theoretical limit. Later work showed that when block space is scarce, the three properties formalized by Roughgarden (2021) cannot all be satisfied exactly at the same time (Chung and Shi, 2023; Chung, Roughgarden, and Shi, 2024). The two papers use different definitions of collusion resistance; the result is an impossibility under both definitions. That is, every transaction fee mechanism must make a trade-off among these three goals.

14.4.5 Automated market maker: the constant-product rule

In a classical exchange, trades occur when buyer and seller orders are matched. In most decentralized exchanges, however, the counterparty is not a person but a **pool**. The pool holds two assets and the price is set by a formula.

Definition 14.11

Constant-product market maker: The pool holds x units of A and y units of B. After every trade the product $x \cdot y = k$ is preserved. A user who sells Δx units of A receives the amount of B that preserves the product:

$$(x + \Delta x) \cdot (y - \Delta y) = x \cdot y \Longrightarrow \Delta y = y \cdot \Delta x / (x + \Delta x)$$

The formula is derived as follows. After the trade the pool holds $x + \Delta x$ of A and $y - \Delta y$ of B, and the product does not change. Divide both sides by $x + \Delta x$: $y - \Delta y = xy/(x + \Delta x)$. Hence $\Delta y = y - xy/(x + \Delta x) = y(x + \Delta x - x)/(x + \Delta x) = y\Delta x/(x + \Delta x)$.

The instantaneous price of the pool is y/x (how many B one A is worth). If the pool is (1000, 1000), the initial price is 1 A = 1 B. But a large trade cannot be made at this price, because as the trade proceeds the B in the pool decreases and B becomes more expensive. The proportion by which the user receives less than at the initial price is called **slippage**.

Δx (A)	Calculation	B received	Slippage
10	$1000 \cdot 10 / 1010$	9.90	%1.0
100	$1000 \cdot 100 / 1100$	90.91	%9.1
250	$1000 \cdot 250 / 1250$	200.00	%20.0
500	$1000 \cdot 500 / 1500$	333.33	%33.3

Table 14.12: For the pool (1000, 1000), the B received and the slippage by amount of A sold.

A general rule follows from the table: slippage equals $\Delta x/(x + \Delta x)$; that is, it depends on the size of the trade relative to the pool. For 10 A one receives 9.90 B instead of 10 B, a slippage of 1%. A user who sells half the pool (500 A) trades at a loss of one third. Real pools also charge a small trading fee. With a 0.3% fee, only 99.7 of 100 A goes into the pool: one receives $1000 \cdot 99.7 / 1099.7 \approx 90.66$ B. The fee stays in the pool and goes to the **liquidity providers** who put assets into the pool.

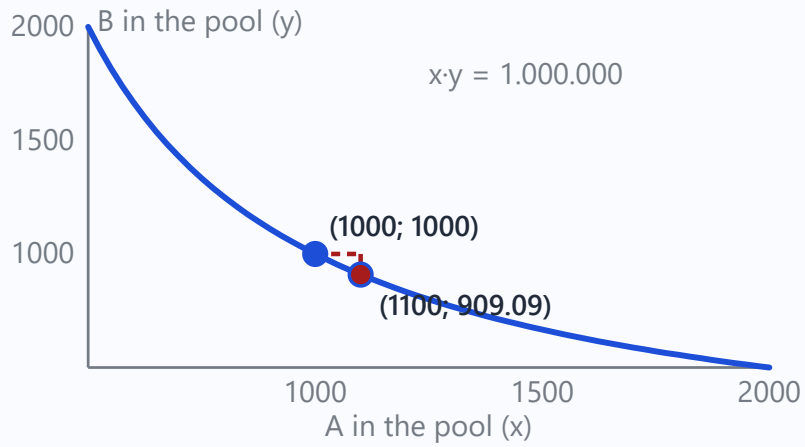


Figure 14.16: The curve $x \cdot y = 1{,}000{,}000$. When 100 A is sold, the pool moves from the point (1000, 1000) to the point (1100, 909.09); as one moves along the curve, B becomes more expensive.

Figure 14.16 shows how a trade moves the pool along the curve. After the 100 A trade the new price is $1100 / 909.09 \approx 1.21$ A: 1 B is now worth 1.21 A. A large trade moves the price against itself; this is called **price impact**. This structure has three consequences:

- When a gap opens between the pool price and the outside market price, **arbitrageurs** who buy cheap and sell dear rebalance the pool. The pool price thereby tracks the outside price.
- The price impact can be computed in advance and exactly by the formula. Where a pending trade will take the pool is known to everyone.
- This is why it is profitable to **get ahead** of a large pending trade (14.4.6).

14.4.6 MEV and the sandwich attack

When the visibility of pending transactions combines with the predictability of price impact, someone who can determine the order of transactions can profit. The best-known way is the **sandwich attack**. The example uses the constant-product pool (1000, 1000); $x \cdot y = 1{,}000{,}000$. A user wants to sell 100 A. Trading alone, the user would receive 90.91 B.

Example 14.9

1. **Seeing.** Before it enters the block, the user's transaction is visible to everyone in the mempool.
2. **Front-running.** The attacker, who can determine the order, puts its own transaction ahead of the user's and sells 50 A. The B it receives: $1000 \cdot 50 / 1050 \approx 47.62$. The pool becomes (1050, 952.38). The attacker has bought B cheaply and pushed the price up.
3. **The user's transaction.** The user sells 100 A: it receives $952.38 \cdot 100 / 1150 \approx 82.82$ B. The pool becomes (1150, 869.57). The user would have received 90.91 B alone; it has lost $90.91 - 82.82 = 8.09$ B. The user's trade pushes the price of B up even further.

4. **Back-running.** The attacker sells its 47.62 B back to the pool. In the pool B is now expensive and A is plentiful: it receives $1150 \cdot 47.62 / (869.57 + 47.62) \approx 59.71$ A. The pool becomes (1090.29, 917.18).
5. **Outcome.** The attacker started with 50 A and ended with 59.71 A: a profit of 9.71 A (excluding trading fees). The user's loss is 8.09 B.

Step	Trade	Pool (A, B)	Result
0	Start	(1000, 1000)	
1	Attacker sells 50 A (front-run)	(1050, 952.38)	receives 47.62 B
2	User sells 100 A	(1150, 869.57)	receives 82.82 B (90.91 alone)
3	Attacker sells the B back (back-run)	(1090.29, 917.18)	receives 59.71 A

Table 14.13: The steps of the sandwich attack, pool $x \cdot y = 1{,}000{,}000$.

The two halves of the attack form a "sandwich": the user's trade lies between the attacker's two trades. The attacker's profit comes from making the user trade at a bad price. The attacker pushes the price of B up before the user, then sells at the price the user has pushed even higher.

Definition 14.12

MEV (maximal extractable value): The value that can be obtained from the power to order, include, and exclude the transactions in a block (Daian et al., 2020).

The sandwich attack is one example of MEV. There are other sources of MEV, such as arbitrage (being the first to close a price gap between pools) and liquidation (being the first to close a loan whose collateral has become insufficient). What they share is that order carries value. From the mechanism design point of view, the basic question is: **who has the ordering power, and who pays the price of this power?** In the sandwich example the user pays.

14.4.7 Code trace: sandwich()

The user has a simple protection: a **slippage tolerance**. The user sets a limit on the least amount of B that will be accepted. If the price falls below this limit, the trade is cancelled. The code below carries out the sandwich attack together with this limit.

```
swap = lambda x, y, dx: y - x * y / (x + dx)  # xy=k
def sandwich(x, y, a, u, min_out):
    b1 = swap(x, y, a)           # front-run: sell a A
    x, y = x + a, y - b1
    b2 = swap(x, y, u)           # user: sell u A
    if b2 < min_out:
        return None              # tx reverted
    x, y = x + u, y - b2
    a2 = swap(y, x, b1)          # back-run: sell b1 B
    return round(a2 - a, 2)
for a in [20, 50, 80]:
    print(a, sandwich(1000, 1000, a, 100, 82))
```

The code runs line by line as follows:

- `swap(x, y, dx)` is the amount of the second asset received when `dx` units of the first asset are sold to the pool: $y - xy/(x + dx)$. This is the same formula as in Definition 14.11.
- `b1`: the attacker sells `a` A and receives `b1` B. The pool is updated.
- `b2`: the user sells `u` A and receives `b2` B. If `b2` is smaller than the user's limit `min_out`, the trade is cancelled and the function returns `None`.
- `a2`: the attacker sells `b1` B back. This time B enters the pool and A leaves it; so the function is called with the arguments swapped: `swap(y, x, b1)`.
- The function returns the attacker's profit, that is, `a2 - a`.

The user will sell 100 A and wants at least 82 B. The attacker tries three values for the front-run amount a.

a	b1 (attacker's B)	b2 (user's B)	b2 ≥ 82?	a2 (A back)	Profit
20	19.61	87.54	yes	24.07	4.07
50	47.62	82.82	yes	59.71	9.71
80	74.07	78.47	no	—	None (cancelled)

Table 14.14: Trace of `sandwich(1000, 1000, a, 100, 82)`.

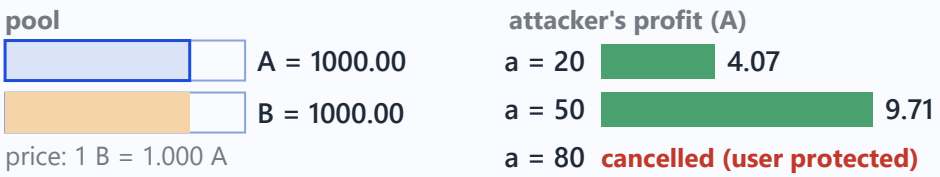


Figure 14.17: The attacker's profit by front-run amount. At $a = 20$ it is 4.07 A, at $a = 50$ it is 9.71 A; at $a = 80$ the user's trade is cancelled and the user is protected.

Figure 14.17 summarizes the result. A larger front-run pushes the price more and brings more profit. But the user's slippage limit cuts this off: at $a = 80$ the user would receive only 78.47 B, so the trade is cancelled and the attacker gains nothing. The attacker's best move is therefore the front-run amount that pushes exactly to the user's limit. A tight tolerance narrows the attacker's room and is a simple protection against MEV.

Common mistake

Thinking a loose slippage tolerance is safe. A loose tolerance raises the chance that the trade goes through, but it leaves the attacker more room to front-run. If the user's limit were 75 B, $a = 80$ would also go through and the attacker would earn more. On the other hand, a very tight tolerance causes the trade to be cancelled even with ordinary price movements. The tolerance is a trade-off.

14.4.8 Countermeasures against MEV

The methods proposed against MEV can be grouped into five ideas. Each has a cost.

Method	Idea	Cost
Tight slippage tolerance	A minimum amount to be received is set; a large front-run causes the trade to be cancelled.	Trades are rejected more often.
Commit–reveal	First a digest (hash) of the transaction is published; the content is revealed once the order is final.	Two steps and delay; transactions that are never revealed.
Encrypted mempool	The transaction is decrypted after its position has been fixed.	Requires trust in whoever holds the key.
Batch auction	All orders in a time slice are matched at the same price; order becomes irrelevant.	Waiting for the length of the slice.
Auctioning the right to order	The right to build the block is sold openly; MEV revenue is distributed competitively rather than secretly.	MEV does not disappear; ordering concentrates in a few large block builders.

Table 14.15: Countermeasures against MEV.

The first three methods restrict either the attacker's information or its room to act. Commit–reveal and the encrypted pool hide the content of the transaction until the order is fixed; the attacker cannot know what to get ahead of. This idea resembles the logic of the sealed-bid auction: bids are opened after everyone has submitted theirs.

A **batch auction** neutralizes the sandwich completely. All buy and sell orders in a time slice are matched at a single price. The attacker's "buy first" and "sell afterwards" orders execute in the same batch as the user's, at the same price. Because the buy and sell prices are the same, the profit is 0. Order no longer carries value.

The last method is a different approach: instead of trying to eliminate MEV, it prices it explicitly. When the right to order is sold in an auction, the value obtainable from MEV is reflected in the price of this right through competition. But this method may lead to power concentrating in a few large actors.

The forms these countermeasures take in practice change often. The lasting idea is this: **ordering power carries value**, and this value must either be reduced (hiding, batch matching) or priced explicitly (auctions). This is a general lesson of mechanism design: if a resource (here, position in the order) is valuable and is distributed for free, a hidden competition arises to capture it.

14.5 Scale and open questions

The examples in the previous sections worked with a few players and a few options. In real applications the numbers of players, items, and strategies are very large. This section addresses the computational problems that scale brings, the approximate methods developed against them, and the open questions of the field.

14.5.1 Computation and scale

The concepts of game theory must be computable. Knowing that an equilibrium exists does not mean that we can find it. Four observations stand out here.

- **Finding an equilibrium can be hard.** Finding a Nash equilibrium in general games is a PPAD-complete problem. That is, no efficient (polynomial-time) algorithm is expected for large games. In practice, **approximate equilibria** and **learning dynamics** are used instead of exact equilibria: players play repeatedly and update their strategies in a way that reduces their regret.
- **Learning and search are powerful together.** Superhuman results in multiplayer poker were obtained by combining regret-minimizing algorithms with search (Brown and Sandholm, 2019). Poker is an imperfect-information, multiplayer, and enormous game; the exact equilibrium cannot be computed, but a good approximate strategy can be found.
- **Combinatorial auctions are hard.** In auctions where several items are sold in bundles, determining the winner is NP-hard. For this reason the bidding language (which bundles bidders can report values for, and how) is designed carefully and approximation algorithms are used.
- **Costs outside the theory matter too.** Latency, data access, and energy costs determine the practicality of a mechanism as much as a theoretical guarantee does. An ad auction must finish within milliseconds; a better but slower rule is useless.

The source of the scale problem is the exponential growth of possibilities. In an auction with m items, each item is either in a bundle or not. The number of subsets is therefore 2^m; removing the empty bundle leaves $2^m - 1$ bundles.

Items m	Number of bundles $2^m - 1$
5	31
10	1.023
20	1.048.575
30	1.073.741.823

Table 14.16: The number of non-empty bundles by number of items.

If every bidder reports a value for every bundle, the input grows exponentially in the number of items. With 30 items the number of values a single bidder would report exceeds one billion. The same explosion appears in games: in a game with n players and two actions each, the number of strategy profiles is 2^n; for $n = 20$ there are 1,048,576 profiles. Even writing a payoff matrix as a table may be impossible. Large games are therefore defined by a short structure, as in congestion games (for example, by one delay function per road).

14.5.2 Winner determination in combinatorial auctions

In a combinatorial auction, bidders bid not on individual items but on bundles of items. For example, a telecommunications firm may want the spectrum licenses of two neighboring regions together; a single region alone is worth little to it. The auctioneer must decide which bids win.

Definition 14.13

Winner determination problem: Each bid consists of a bundle and a value: (S_i, v_i). The problem is to choose, from among bids that share no items (do not overlap), the set that maximizes the total value.

This problem is the same as the **weighted set packing** problem of computer science. It is NP-hard and in general cannot be well approximated: it cannot be guaranteed that an efficient algorithm will stay within a given factor of the optimum on every input. The brute-force method tries 2^n subsets for n bids. In practice, integer programming solvers are used that prune this search cleverly (cutting unpromising branches early).

The hardness of winner determination also affects the VCG mechanism. In VCG each winner pays the harm its presence causes the others. To compute this, the problem is solved once with all the bids and then once with each bidder removed; a total of $n + 1$ solutions are needed. If the problem is solved only approximately, the truthfulness guarantee of VCG can break. A bidder can lie to exploit the error of the approximation algorithm and obtain a better allocation in its own favor.

Example 14.10

There are three items (a, b, c) and four bids.

Bid	Bundle	Value
t1	{a, b, c}	10
t2	{a}	4
t3	{b}	4
t4	{c}	3

Table 14.17: Four bundle bids.

1. t1 contains all the items, so it overlaps with the other three bids. If t1 is chosen, no other bid can be chosen: total 10.
2. t2, t3, and t4 do not overlap with one another. They can all be chosen together: *4 + 4 + 3 = 11*.
3. Intermediate choices are worth less: for example {t2, t3} = 8.
4. The optimum is {t2, t3, t4} = 11. A single large bid is worth less than the sum of the small bids.

14.5.3 Code trace: greedy()

A common approach for NP-hard problems is **greedy** selection: the bids are sorted by a key and, in order, every bid that does not overlap with the earlier ones is taken. What determines the result is which key is used.

```
def greedy(bids, key):
    ranked = sorted(bids, key=key, reverse=True)
    total, used = 0, set()
    for S, v in ranked:
        if not (S & used):        # no overlap
            total += v
            used |= S
    return total
T = [({'a','b','c'}, 10), ({'a'}, 4), ({'b'}, 4), ({'c'}, 3)]
print(greedy(T, lambda t: t[1]))
print(greedy(T, lambda t: t[1] / len(t[0])))
```

The code runs line by line as follows. `ranked` sorts the bids from largest to smallest by the given key. `total` holds the value of what has been selected, and `used` holds the items in the selected bundles. For each bid, the loop looks at the intersection of the bundle with the used items (`S & used`). If the intersection is empty, the bid is taken: its value is added to the total and its items are added to the used set (`|=`). The code is run with two keys: first the value of the bid, then the value per item.

Step	By value	Total	By value per item	Total
1	t1 {a, b, c} (10): taken	10	t2 {a} (4): taken	4

Step	By value	Total	By value per item	Total
2	t2 {a} (4): a is used, skipped	10	t3 {b} (4): taken	8
3	t3 {b} (4): b is used, skipped	10	t1 {a, b, c} (10/3 ≈ 3.33): a and b are used, skipped	8
4	t4 {c} (3): c is used, skipped	10	t4 {c} (3): taken	11

Table 14.18: Trace of `greedy()`; left: by value, right: by value per item.

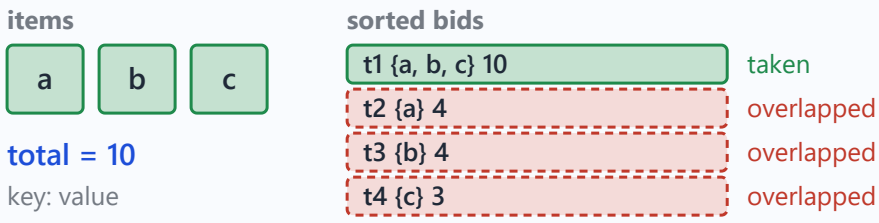


Figure 14.18: Greedy selection by value. t1 is taken first and fills all three items; the other three bids overlap with it. Total 10.

Figure 14.18 shows the result of the first key. Sorting by value takes the large bundle early and leaves out all the small bids. The result is 10; the optimum is 11. Greedy selection by value is fooled by large bundles, because it does not compare the value of a bundle with the resources it uses.

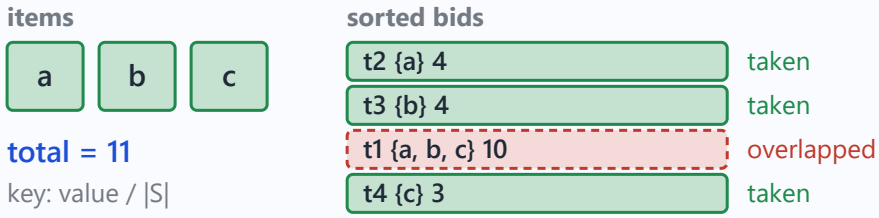


Figure 14.19: Greedy selection by value per item. t2 and t3 are taken first; t1 is skipped; t4 takes the free item c. Total 11 = optimum.

In Figure 14.19 the second key is used. The values per item are 4 for t2 and t3, *10/3 ≈ 3.33* for t1, and 3 for t4. t2 and t3 are tied; Python's `sorted` function keeps the original order on ties. Because this ordering measures value relative to the resources, it finds the optimum.

But this key can also fail. Consider the bids ({a, b}, 10) and ({a}, 6). The values per item are 5 and 6; greedy selection takes the 6 first, then skips the 10 because it overlaps, and stays at 6. Yet the bid of 10 alone is better. Both keys have bad examples. Lehmann, O'Callaghan, and Shoham (2002) proposed a key that divides the value by the square root of the bundle size: $v / \sqrt{|S|}$. With this key, greedy selection guarantees, even in the worst case, at least $1/\sqrt{m}$ of the optimal value (m is the number of items). Moreover, combined with a suitable payment rule, this greedy algorithm is incentive compatible when each bidder values only a single bundle (*single-minded* bidders).

Common mistake

Assuming that a greedy rule that finds the optimum in one example is always right. Value per item finds the optimum in the example of Table 14.17 but not in the example of ({a, b}, 10) and ({a}, 6). For an NP-hard problem, a simple rule is meaningful only together with an approximation guarantee.

14.5.4 Other research directions

Besides the main topics of the chapter, four more research directions stand out.

Influence maximization in social networks. A firm will give a free product to k people and hopes that they will spread it to their friends. Who are the k "seed" people that maximize the spread (Kempe, Kleinberg, and Tardos, 2003)? The spread is described by a model.

Definition 14.14

Independent cascade model: Initially the seed people are influenced. Each newly influenced person tries to influence each of its neighbors exactly once, with probability p. The influenced neighbors do the same in the next step. The process stops when nobody is newly influenced.

Finding the best seed set is NP-hard. But the problem has a structure: the expected spread is **submodular**, that is, it has diminishing marginal returns. The additional contribution of making a person a seed decreases as the seed set grows, because the people that person would reach may already have been reached by other seeds. Thanks to this structure a simple greedy algorithm works well: at each step the person who contributes most to the expected spread is added to the seed set. This algorithm guarantees at least $1 - 1/e \approx 63\%$ of the best spread. Unlike the winner determination problem, here the structure gives greedy selection a strong guarantee.

Multi-agent reinforcement learning
: Each agent's environment includes the other learning agents. Because the others also change as one agent learns, the environment is not stationary. Convergence guarantees in single-agent learning assume that the environment is fixed; so they do not hold here. Algorithmic collusion is an example of this problem.

Sustainable systems
: Mechanisms such as carbon emission permit markets and demand response in electricity price an externality. The cost a player imposes on others is charged to that player through a tax (a Pigouvian tax). This is the same idea as in congestion pricing.

Robustness
: Mechanisms that work well despite adversarial, faulty, or boundedly rational players. Classical theory assumes that everyone is rational; in real systems some participants do not conform to this assumption.

In addition, the following directions are active: **information design** or Bayesian persuasion (steering the receiver's decision by choosing the rule for disclosing information); aggregating human preferences in AI alignment (producing a single objective from the preferences of many people is a social choice problem); and **prediction markets** and **peer prediction** methods, which reward unverifiable reports by comparing them with the reports of others.

14.5.5 Open research questions

The questions the field is seeking answers to today can be grouped under six headings. These questions rest not on a particular technology but on lasting tensions seen throughout the chapter.

Human + AI markets
: How should the rules change when humans and agents interact in the same market? Agents are fast and tireless; humans are slow and have limited attention. The same rule can affect the two groups differently.

Quantifying trade-offs
: How should the trade-offs among efficiency, privacy, and fairness be measured? In the exponential mechanism, privacy increases as ε decreases but revenue falls; there is no common language for such trade-offs.

Adaptive robustness
: How can strategy-robust mechanisms adapt to changing behavior? If a mechanism adapts by learning, the learning process itself can be manipulated.

Theory and data
: How can the assumptions of models be tested with real platform data? Theory assumes particular value distributions and rationality; the data may not confirm these assumptions.

Equilibrium of learning agents
: What do algorithms that learn against one another converge to, and how can collusion be prevented?

Verifiable mechanisms
: How can it be proved or audited that a learned rule is incentive compatible? For hand-designed rules there are proofs; for neural network rules there is only measured regret.

14.6 Review: common mistakes and worked problems

This section collects the chapter's numerical results in a table, lists the most common mistakes together, and gives the example problems from the slides with their solutions.

Topic	Description	Number in this chapter
Bargaining between agents	Backward induction; patience and the last word are power	$T = 4$: A 18.1; infinite horizon 52.6
Algorithmic collusion	The trigger strategy sustains cooperation if δ is large enough	$\delta \geq 1/2$

Topic	Description	Number in this chapter
Price from data	Empirical revenue maximization; overfitting with few samples	$p = 0.47$: empirical 0.294, true 0.249
Exponential mechanism	Probability $\propto exp(\varepsilon \cdot revenue / 2\Delta)$; privacy + approximate IC	$\varepsilon = 2$: ratio $\le e^2 \approx 7.39$ (1.44 in the example)
Base fee	+12.5% for a full block, −12.5% for an empty one; the fee is burned	100 → 160.2 → 115
MEV	Ordering power carries value; the sandwich attack	User −8.09 B, attacker +9.71 A
Course map	Equilibrium → design → application → learning → future	A single concept map from the preliminaries to future directions

Table 14.19: The topics of the chapter and their key numbers.

Common mistake

- **Forgetting the number of firms:** The patience that suffices with two firms is not enough in a crowded market: the threshold is $\delta \ge 1 - 1/n$, which is 0.80 for $n = 5$.
- **Confusing finite and infinite horizons:** With a finite horizon the last proposer is strong (A gets 18.1 at $T = 4$); with an infinite horizon the first proposer is slightly ahead (52.6).
- **Taking empirical revenue for the truth:** The success of a price selected and measured on the same data is overstated; separate validation data is needed.
- **Linear proportion in the exponential mechanism:** The probability is proportional not to revenue but to $exp(\varepsilon \cdot R / 2\Delta)$; a revenue difference translates exponentially into the probability ratio.
- **Updating the base fee additively:** The change is multiplicative: two full blocks give 100 → 112.5 → 126.6 (not 125). The base fee does not go to the producer; it is burned.
- **Thinking a loose slippage tolerance is safe:** It raises the chance that the trade goes through, but leaves the attacker more room to front-run.

The six worked problems below briefly repeat the chapter's calculations.

Example 14.11

1. **Two-round bargaining.** 100 TL, A proposes and then B, $\delta = 0.9$. Round 2 is the last round and B gets 100 there. In round 1, A offers B its waiting value: $0.9 \cdot 100 = 90$. The division is A 10, B 90. In the infinite horizon A's share is $100/1.9 \approx 52.6$.

2. **Trigger with three firms.** The monopoly profit of 100 is shared equally; a deviator gets 100 for one period, then 0. The condition is $(100/3)/(1 - \delta) \ge 100 \iff 1 - \delta \le 1/3 \iff \delta \ge 2/3$. As the number of players grows, collusion gets harder.

3. **Empirical price.** The sample values are 0.9, 0.6, 0.5, 0.2. The revenues are $0.9 \cdot 1/4 = 0.225$, $0.6 \cdot 2/4 = 0.3$, $0.5 \cdot 3/4 = 0.375$, $0.2 \cdot 4/4 = 0.2$. The best is $p = 0.5$, with revenue 0.375.
4. **Base fee.** $r = 100$, three full blocks and one empty block: $100 \cdot 1.125^3 \cdot 0.875 \approx 124.58$.
5. **Map question.** In the GSP–VCG comparison at the Online Platforms node, VCG comes from the Mechanism Design topic. The second-price idea on which GSP rests comes from the second-price (Vickrey) auction in the Auctions topic.
6. **Exponential mechanism.** Two candidate prices have revenues 2 and 1, $\varepsilon = 2$, $\Delta = 1$. The weights are e^2 and e^1. The probabilities are $e^2/(e^2 + e) = e/(e + 1) \approx 0.731$ and $1/(e + 1) \approx 0.269$.

One simple example to try on your own is the pizza vote: three friends must agree on a single pizza, each proposes toppings, and then they vote. Even a small console application shows the voting rule and strategic voting (14.6.1).

14.6.1 Example: the pizza vote

A small example can show modeling a situation, solving it, and seeing strategic behavior all at once. Three friends must agree on a single pizza. Their preferences are as follows:

Person	1st	2nd	3rd
Ali	mushroom	sausage	olives
Banu	sausage	mushroom	olives
Cem	olives	sausage	mushroom

Table 14.20: The true preferences in the pizza vote.

Example 14.12

1. **Model.** The players are the three people. A strategy is the reported ranking. The payoff is 2, 1, or 0 depending on where the winning topping stands in the person's true ranking.
2. **Plurality.** Each candidate gets one first-place vote: a 1–1–1 tie. The plurality rule cannot decide.
3. **Borda (2, 1, 0).** Mushroom: $2 + 1 + 0 = 3$; sausage: $1 + 2 + 1 = 4$; olives: $0 + 0 + 2 = 2$. Sausage wins.
4. **Condorcet.** Sausage versus mushroom: Banu and Cem prefer sausage, 2–1. Sausage versus olives: Ali and Banu prefer sausage, 2–1. Sausage wins both pairwise comparisons: it is the Condorcet winner.

5. **Strategic vote.** If Ali buries sausage at the bottom and reports "mushroom, olives, sausage", the Borda scores become mushroom $2 + 1 + 0 = 3$, olives $1 + 0 + 2 = 3$, sausage $0 + 2 + 1 = 3$. If the tie is broken alphabetically, mushroom wins. By lying, Ali has gotten his favorite chosen: his payoff rises from 1 to 2.

This finding is an example of the **burying** strategy in the Borda count (pushing the strong rival to the bottom of the ranking) and a small reflection of the Gibbard–Satterthwaite theorem. The example can also be tried in code. A console version takes the rankings and counts them under the plurality, two-round runoff, Borda, Condorcet, and approval voting rules side by side. In a web version the user changes one player's report and sees how the winner shifts. Changing the tie-breaking rule also changes the winner; a small detail of the rule can determine the outcome.

14.7 Summary

- Software agents are now players: they have goals, knowledge, and strategies. The principal–agent problem, sensitivity to framing, prompt injection, and accountability are game-theoretic questions that regain importance with agents.
- Finite-horizon alternating-offers bargaining is solved by backward induction; the last word is power. With 100 TL, $\delta = 0.9$, and four rounds, the first proposer gets only 18.1.
- In the infinite horizon the Rubinstein solution gives the first proposer the share $(1 - \delta_B)/(1 - \delta_A\delta_B)$; with symmetric patience, $1/(1 + \delta)$. Patience is bargaining power.
- The grim trigger strategy sustains collusion if $\delta \geq 1 - 1/n$. Frequent price updates, transparent prices, and price-matching guarantees facilitate collusion; learning pricing algorithms can learn a high price without an explicit agreement.
- There is no universally best strategy: an agent's success depends on the game, the opponents, noise, and memory.
- Automated mechanism design finds the rule as the solution of an optimization with IC and IR constraints. In the two-type example the optimum is a vertex of the linear program and corresponds to a posted price.
- Empirical revenue maximization overstates its success because it is selected and measured on the same data (0.294 against a true 0.249). Sample complexity asks how many samples a good price requires; separate data is needed for selection and for measurement.
- The exponential mechanism selects candidates with probability proportional to $\exp(\varepsilon \cdot R / 2\Delta)$; it provides ε-differential privacy and approximate incentive compatibility.
- A smart contract provides commitment through code but makes pending transactions visible. The base fee + tip rule simplifies bidding for users; the base fee changes by at most 12.5% per block according to demand and is burned to make fake demand expensive.
- The three goals of user incentive compatibility, producer incentive compatibility, and collusion resistance cannot all be fully satisfied together when block space is scarce.

- In a constant-product market maker the price impact can be computed in advance; this makes the sandwich attack possible. MEV is the value of ordering power; countermeasures either reduce this value or price it explicitly. A tight slippage tolerance is the simplest protection.
- Winner determination in combinatorial auctions is NP-hard; greedy rules are approximate and the choice of key determines the result. In influence maximization, submodularity gives greedy selection a $1 - 1/e$ guarantee.

14.8 Key terms

principal–agent problem framing effect prompt injection backward induction
subgame perfect equilibrium discount factor Rubinstein bargaining model stationary strategy
collusion algorithmic collusion grim trigger strategy reinforcement learning
automated mechanism design approximate incentive compatibility empirical revenue overfitting
sample complexity cross-validation differential privacy exponential mechanism smart contract
commitment mempool transaction fee mechanism base fee tip fee burning
automated market maker slippage tolerance price impact maximal extractable value
sandwich attack front-running back-running batch auction combinatorial auction
winner determination problem NP-hard PPAD-complete influence maximization
independent cascade model submodularity multi-agent reinforcement learning Pigouvian tax
information design prediction market peer prediction reproducibility confidence interval
Borda count Condorcet winner

14.9 Test Yourself

1. Two agents will share 100 TL over three rounds: A proposes in rounds 1 and 3, B in round 2; $\delta = 0.8$. What is the division in the equilibrium found by backward induction? In the infinite horizon, if A's patience were $\delta_A = 0.8$ and B's were $\delta_B = 0.9$, what would A's share be?

   Answer

   In round 3 the last word is A's: A gets 100. In round 2 B offers A its waiting value $0.8 \cdot 100 = 80$: A 80, B 20. In round 1 A offers B $0.8 \cdot 20 = 16$: A 84, B 16, and the proposal is accepted immediately. In the infinite horizon $x^* = (1 - 0.9)/(1 - 0.72) = 0.1/0.28 \approx 0.357$: A gets 35.7. The more impatient A gets the small share even though it is the first proposer.

2. Two firms each earn a profit of 50 per period at the monopoly price. A firm that undercuts gets 100 per period; however, the rival notices the undercutting only two periods later, so the deviator gets 100 for two periods and then 0 forever. For which δ does the grim trigger strategy sustain

cooperation? How does the threshold change compared with immediate detection?

Answer

Cooperation is worth $50/(1-\delta)$ and deviation is worth $100+100\delta$. The condition is $50/(1-\delta) \geq 100(1+\delta) \iff 50 \geq 100(1-\delta)(1+\delta) = 100(1-\delta^2) \iff \delta^2 \geq 1/2 \iff \delta \geq 0.707$. With immediate detection the threshold was 0.5. Delayed detection makes deviation more profitable and raises the required patience; this is why transparent prices facilitate collusion.

3. The 5 buyer values observed in the past are 0.8, 0.7, 0.3, 0.25, 0.1. Which price maximizes the empirical revenue, and what is that revenue? If the values actually come from $U[0, 1]$, what is the true revenue of this price?

Answer

The revenues are $0.8 \cdot 1/5 = 0.16$, $0.7 \cdot 2/5 = 0.28$, $0.3 \cdot 3/5 = 0.18$, $0.25 \cdot 4/5 = 0.2$, $0.1 \cdot 5/5 = 0.1$. The best is $p = 0.7$, with empirical revenue 0.28. The true revenue is $0.7 \cdot 0.3 = 0.21$, below the best 0.25. Five samples have pushed the price away from the true best (0.5), and the empirical revenue is again optimistic.

4. In an exponential mechanism the scores of three candidate prices are 0, 1, and 2; $\varepsilon = 1$, $\Delta = 1$. What are the selection probabilities? How many times the probability of the worst candidate is the probability of the best?

Answer

The weights are $\exp(R/2)$: $e^0 = 1$, $e^{0.5} \approx 1.649$, $e^1 \approx 2.718$; the total is ≈ 5.367. The probabilities are ≈ 0.186, 0.307, 0.506. The best candidate gets $e^1 \approx 2.72$ times the probability of the worst; this has nothing to do with the score ratio (2/0).

5. The base fee is 200. The next four blocks are 100%, 100%, 0%, 0% full. What is the base fee after the fourth block? Why does the fee not return to 200?

Answer

$200 \to 200 \cdot 1.125 = 225 \to 253.125 \to 253.125 \cdot 0.875 \approx 221.48 \to \approx 193.80$. The change is multiplicative and $1.125 \cdot 0.875 \approx 0.984 < 1$; an equal number of full and empty blocks lowers the fee slightly.

6. The constant-product pool is (2000, 2000). A user sells 200 A. How much B does it receive and what is the slippage in percent? How does this compare with selling 100 A in the (1000, 1000)

pool?

Answer

$\Delta y = 2000 \cdot 200 / 2200 \approx 181.82$ B; slippage $1 - 181.82/200 \approx 9.1\%$. Selling 100 A in the (1000, 1000) pool also gives 9.1% slippage. Slippage depends on $\Delta x/(x + \Delta x)$, that is, on the size of the trade relative to the pool; a deep pool carries the same trade with less slippage.

7. Four bundle bids: t1 = ({a, b}, 6), t2 = ({b, c}, 6), t3 = ({a}, 4), t4 = ({c}, 4). What totals do greedy selection by value and by value per item give? What is the optimum?

Answer

By value the order is t1, t2, t3, t4: t1 is taken (a, b); t2 is skipped because of b, t3 because of a; t4 is taken. Total 10. The value per item is 4 for t3 and t4 and 3 for t1 and t2: t3 and t4 are taken, and t1 (a) and t2 (c) overlap. Total 8. The optimum is 10 ({t1, t4} or {t2, t3}). This time value per item is fooled by the small bundles.

8. In an agent tournament a pairing is repeated 25 times; the mean score is 3.1 and the standard deviation 0.5. What is the approximate 95% confidence interval? How does the width of the interval change if the number of repetitions is raised to 100?

Answer

The half-width is $1.96 \cdot 0.5 / \sqrt{25} = 1.96 \cdot 0.1 \approx 0.196$; the interval is about [2.90, 3.30]. With 100 repetitions $\sqrt{100} = 10$ and the half-width falls to ≈ 0.098: four times the repetitions halve the interval.

14.10 Further reading

- T. Roughgarden, *Twenty Lectures on Algorithmic Game Theory* (2016): covers mechanism design, combinatorial auctions, and learning dynamics from a computer science perspective, in short lectures.
- T. Roughgarden, "Transaction Fee Mechanism Design" (2021): defines the goals of transaction fee mechanisms formally (user and producer incentive compatibility, collusion resistance) and analyzes the base fee rule against these goals.
- E. Calvano, G. Calzolari, V. Denicolò, S. Pastorello, "Artificial Intelligence, Algorithmic Pricing, and Collusion" (2020): gives detailed simulations in which learning pricing agents learn a high price without communicating and punish deviations.
- D. Easley, J. Kleinberg, *Networks, Crowds, and Markets* (2010): explains spreading in networks, bargaining, and markets with little mathematics and many examples.

14.11 References

1. R. Axelrod, “Effective choice in the Prisoner's Dilemma”, *Journal of Conflict Resolution* 24(1), 1980.
2. N. Brown, T. Sandholm, “Superhuman AI for multiplayer poker”, *Science*, 2019.
3. E. Calvano, G. Calzolari, V. Denicolò, S. Pastorello, “Artificial Intelligence, Algorithmic Pricing, and Collusion”, *American Economic Review*, 2020.
4. H. Chung, E. Shi, “Foundations of transaction fee mechanism design”, *ACM-SIAM SODA*, 2023.
5. H. Chung, T. Roughgarden, E. Shi, “Collusion-resilience in transaction fee mechanism design”, *ACM EC*, 2024.
6. R. Cole, T. Roughgarden, “The Sample Complexity of Revenue Maximization”, *ACM STOC*, 2014.
7. V. Conitzer, T. Sandholm, “Complexity of Mechanism Design”, *UAI*, 2002.
8. P. Daian et al., “Flash Boys 2.0: Frontrunning in Decentralized Exchanges, Miner Extractable Value, and Consensus Instability”, *IEEE S&P*, 2020.
9. P. Dütting, Z. Feng, H. Narasimhan, D. Parkes, S. Ravindranath, “Optimal Auctions through Deep Learning”, *ICML*, 2019.
10. D. Easley, J. Kleinberg, *Networks, Crowds, and Markets*, Cambridge University Press, 2010.
11. R. A. Hearn, E. D. Demaine, *Games, Puzzles, and Computation*, A K Peters, 2009.
12. A. R. Karlin, Y. Peres, *Game Theory, Alive*, AMS, 2017.
13. D. Kempe, J. Kleinberg, É. Tardos, “Maximizing the Spread of Influence through a Social Network”, *KDD*, 2003.
14. D. Lehmann, L. I. O'Callaghan, Y. Shoham, “Truth revelation in approximately efficient combinatorial auctions”, *Journal of the ACM* 49(5), 2002.
15. F. McSherry, K. Talwar, “Mechanism Design via Differential Privacy”, *IEEE FOCS*, 2007.
16. N. Nisan, T. Roughgarden, É. Tardos, V. Vazirani (ed.), *Algorithmic Game Theory*, Cambridge University Press, 2007.
17. T. Roughgarden, *Twenty Lectures on Algorithmic Game Theory*, Cambridge University Press, 2016.
18. T. Roughgarden, “Transaction Fee Mechanism Design”, *ACM EC*, 2021.
19. A. Rubinstein, “Perfect Equilibrium in a Bargaining Model”, *Econometrica* 50(1), 1982.

APPENDIX A

Glossary of Terms

The terms used in the book, each with a short definition. The number next to a term is the chapter where it is explained in detail. A searchable version of the glossary is on the course page.

#

(agent) goal alignment 14
The condition that the objectives of an AI agent are consistent with those of the people or system it serves.

2-approval 7
A voting rule in which each voter approves of up to two candidates and the candidate with the most approvals wins.

A

accountability 11
The property that those who build or deploy an algorithm can be identified and held answerable for its outcomes.

accuracy 11
The fraction of a classifier's predictions that are correct.

action 2
One of the choices available to a player at a decision point in a game.

adaptation 13
The process by which an agent changes its behavior in response to feedback or a changing environment.

adaptive 14
Describing a method that changes its behavior according to what it has observed so far.

additivity 10
The property that the value of a union of disjoint parts equals the sum of their values.

adoption model 6
A model describing how a new product, technology, or behavior spreads through a population over time.

advertiser 6
A party that pays a platform to display its advertisements, typically through a sponsored-search auction.

affine transformation 1
A transformation of the form x maps to ax + b, which rescales and shifts utilities without changing the preferences they represent.

agenda (manipulation) 7
A manipulation of a voting process in which the order of pairwise votes is chosen to bring about a desired winner.

agent 10
An entity that perceives its environment and acts on it in pursuit of its own goals.

agent architecture 10
The internal structure of an agent, describing how it perceives, decides, and acts.

AI agent 14
A software system that perceives its environment, makes decisions, and acts autonomously, often using machine learning.

algorithmic audit 11
An independent examination of an algorithm's inputs, outputs, and effects to detect bias or other harms.

algorithmic collusion 14
Anti-competitive coordination between pricing or bidding algorithms that leads to supracompetitive outcomes without explicit communication.

algorithmic fairness 11
The study and design of algorithms whose decisions treat individuals and groups equitably according to a formal fairness criterion.

algorithmic game theory (AGT) 2
The field that combines game theory and computer science to analyze strategic behavior and design algorithms and mechanisms for it.

all-pay auction 5
An auction in which every bidder pays their bid regardless of whether they win.

Allais paradox 1
The observed violation of the independence axiom, in which people's choices between lotteries depend on how a common outcome is framed.

allocation 4
The outcome of a mechanism that determines who receives which items or resources.

allocation rule 5
The part of a mechanism that maps the reported bids or preferences to an allocation of items.

alpha–beta pruning 2
A search optimization that skips branches that cannot affect the minimax value, giving the same result with far fewer nodes examined.

alternating-offers bargaining 14
A bargaining model in which two players take turns proposing how to divide a surplus until one accepts.

altruistic punishment 12
Willingness to pay a cost to punish those who violate cooperative norms, even without personal gain.

ambiguity aversion 1
A preference for known risks over unknown risks, as shown in the Ellsberg paradox.

anonymity 7
The requirement that a voting rule treat all voters the same, so that renaming voters does not change the outcome.

anti-coordination 2
A game in which players do best by choosing different actions from one another.

antisocial punishment 12
Punishment directed at cooperators rather than at free riders.

approval voting 7
A voting system in which each voter may approve any number of candidates and the candidate with the most approvals wins.

approximate incentive compatibility 14
A relaxed guarantee that truthful reporting is within a small additive or multiplicative amount of the best possible response.

approximation algorithm 9
An algorithm that runs in polynomial time and returns a solution with a provable bound on its distance from the optimum.

approximation ratio 9
The worst-case ratio between the value of an approximate solution and the value of an optimal solution.

argmax 0
The operator that returns the argument at which a function attains its maximum value.

Arrow's impossibility theorem 7
The theorem that no ranking rule with three or more alternatives can satisfy unanimity, independence of irrelevant alternatives, and non-dictatorship together.

ask (sell order, double auction) 5
An offer to sell a good at a stated minimum price in a double auction.

asymmetric game 2
A game in which players have different action sets or payoffs, so that swapping their roles changes the game.

atomic 8
Describing a routing model in which each player controls a single indivisible unit of traffic that must use one path.

auction 5
A mechanism for allocating goods and determining prices based on bids submitted by participants.

automated market maker (AMM) 14
An algorithmic protocol that sets prices for trading assets from a pool of liquidity, without an order book.

automated mechanism design 14
The use of computational search or learning to find a mechanism that optimizes a given objective subject to incentive constraints.

autonomy 10
The ability of an agent to act on its own initiative without direct external control.

average mechanism 4
A mechanism whose outcome is the average of the reported values or positions.

Axelrod tournament 13
A tournament in which computer programs played the repeated Prisoner's Dilemma, won by the simple strategy Tit for Tat.

B

back-running 14
Placing a transaction right after a target transaction in a block in order to profit from its effect.

backward induction 2
A method of solving a sequential game by reasoning from the final decisions back to the first.

Balance algorithm 6
An online algorithm for budgeted ad allocation that bids using a function of how much of each advertiser's budget has been spent.

balance for the positive/negative class 11
A fairness criterion requiring that the average score given to actual positives (or negatives) be the same across groups.

balanced game 10
A cooperative game whose core is non-empty.

ballot 7
The document on which a voter records their preferences or choices.

bandwidth 8
The capacity of a link, expressed as the maximum rate of data it can carry.

Banzhaf index 10
A power index that measures how often a player is a critical member of a winning coalition.

bargaining 14
A process in which parties negotiate to agree on how to divide a surplus or settle terms.

base fee 14
The minimum per-unit price that every transaction in a block must pay under a fee mechanism such as EIP-1559.

base rate 11
The overall frequency of a class in a population, regardless of any individual evidence.

base-rate neglect 1
The cognitive bias of ignoring general frequencies and giving too much weight to case-specific evidence.

batch auction 14
An auction that collects orders over a period and clears them all together at a single price.

Battle of the Sexes 2
A coordination game in which two players prefer to be together but each prefers a different joint activity.

Bayes' rule 0
The rule that updates the probability of a hypothesis after observing evidence, computed as likelihood times prior divided by the evidence probability.

Bayesian average 6
A rating that blends an item's own average with the overall average, weighted by the number of ratings it has.

Bayesian decision rule 1
The rule that chooses the action with the highest expected utility under the posterior belief.

Bayesian game 4
A game in which players have private information about their types and hold beliefs about the types of the others.

Bayesian incentive compatible (BIC) 4
A property of a mechanism in which truthful reporting is a Bayes–Nash equilibrium.

Bayes–Nash equilibrium (BNE) 4
A strategy profile in which each player's strategy maximizes their expected payoff given their type and their beliefs about the others.

Beckmann potential 8
A potential function for congestion games whose minimizers are Nash equilibria.

belief–desire–intention (BDI) 10
An agent architecture that represents the agent's mental state through beliefs, desires, and intentions.

Bell number 10
The number of ways to partition a set of n elements into non-empty subsets; it counts possible coalition structures.

Bellman–Ford algorithm 8
A shortest-path algorithm that handles negative edge weights by repeatedly relaxing all edges.

Bertrand competition 3
A model of price competition in which firms simultaneously set prices for identical goods.

Bertrand paradox 3
The result that two firms in Bertrand competition end up pricing at marginal cost and earning zero profit.

best response 3
A strategy that gives a player the highest payoff against the given strategies of the others.

best-response correspondence 3
The map that assigns to each profile of opponents' strategies the set of the player's best responses.

best-response curve 3
A graph of a player's best response as a function of the other player's strategy; equilibria lie at the intersections of the curves.

best-response dynamics 13
A learning process in which players repeatedly switch to a best response to the current strategies of the others.

best-response marking 2
The technique of marking each player's best-response payoffs in a payoff matrix to find pure Nash equilibria.

BGP (Border Gateway Protocol) 8
The inter-domain routing protocol of the Internet, with which autonomous systems exchange reachability information.

bias (societal / data) 11
Systematic distortion in data or algorithms that produces unfair outcomes for some people or groups.

bid 5
The amount a participant offers to pay for a good in an auction.

bid (buy order, double auction) 5
An offer to buy a good at a stated maximum price in a double auction.

bid shading 5
Submitting a bid below one's true value in order to gain positive surplus from winning.

bidder 5
A participant who submits bids in an auction.

big-O and Θ notation 0
Notation that describes how a quantity grows with input size, with O giving an upper bound and Θ a tight bound.

bilateral trade 4
Trade between one buyer and one seller, each with private valuations.

binding agreement 2
An agreement that the parties are bound to honor, so that deviations can be enforced.

bisection 11
A method of finding a root or optimum by repeatedly halving an interval that contains it.

block producer 14
The participant who chooses which transactions to include in the next block and in what order.

blockchain 14
A distributed, append-only ledger maintained by many participants without a central authority.

blocking pair 4
A pair of agents who would both prefer each other to their assigned partners, making the matching unstable.

bluffing 1
Misleading opponents about the strength of one's hand by betting or acting as though it were stronger or weaker.

Bondareva–Shapley theorem 10
The theorem that a cooperative game has a non-empty core if and only if it is balanced.

Borda count 7
A voting rule in which candidates receive points according to their rank on each ballot and the highest total wins.

Boston mechanism (immediate acceptance) 4
A school-choice mechanism that assigns seats in rounds and fixes a seat as soon as a school's applicant ranks it highly enough.

bounded rationality 12
The idea that decision makers have limited information, time, and computational ability and so do not optimize perfectly.

BPR latency function 8
A latency function of the form t0 times one plus a constant times the load-to-capacity ratio raised to a power, used for road traffic.

Braess's paradox 8
The paradox that adding a road to a network can increase the travel time of every driver at equilibrium.

branch 2
A path from a node to one of its children in a game tree, representing a single move.

branching factor 2
The average number of moves available at each node of a game tree.

budget balance 4
The requirement that a mechanism's payments cover its costs, with no money created or lost.

budget surplus 4
The amount by which total payments collected exceed what must be paid out.

bundle 11
A set of items that a bidder values and is allocated together.

burying 7
A manipulation in a voting system in which voters rank a strong rival low to defeat them.

buyer / seller surplus 5
The difference between what a buyer is willing to pay and what they pay, and between the price a seller receives and their cost.

C

cake cutting 11
The problem of fairly dividing a heterogeneous divisible resource among several agents.

calibration (within groups) 11
The property that among people given the same score, the share who truly belong to the positive class is the same across groups.

capacity 8
The maximum amount of flow or traffic that an edge or system can carry.

capacity augmentation 9
A technique that evaluates a system's performance when its capacity is increased by a factor, to bound the loss from selfishness.

cardinal (von Neumann–Morgenstern) utility 1
A utility scale derived from preferences over lotteries, unique up to positive affine transformations.

cartel 3
A group of firms that coordinate prices or output to restrict competition.

Cartesian product 0
The set of all ordered pairs formed by taking one element from each of two sets.

case analysis 2
A proof or solution technique that divides the problem into exhaustive cases and treats each separately.

centipede game 2
A sequential game in which two players alternately choose to continue or stop, with growing payoffs that make stopping early rational by backward induction.

certainty effect 1
The tendency to overweight outcomes that are certain relative to those that are merely probable.

certainty equivalent 1
The guaranteed amount that a decision maker values the same as a given risky lottery.

chain rule 0
A rule for differentiating a composition of functions as the product of the derivatives.

chance node 1
A node in a game or decision tree where nature determines the next branch randomly.

characteristic function 10
A function that gives the value that each coalition of players can achieve on its own.

chicken-and-egg problem 6
The difficulty of attracting either side of a platform without already having the other side.

Chouldechova's identity 11
An identity relating false positive rate, false negative rate, prevalence, and positive predictive value, showing they cannot all be equal across groups.

Clarke pivot rule 4
A rule in the VCG mechanism that sets each agent's payment to the harm their participation causes to the others.

click-through rate (CTR) 6
The fraction of times an ad is clicked when shown.

coalition 10
A subset of the players of a game who coordinate their actions.

coalition formation 10
The process by which agents decide which groups to form and with whom to cooperate.

coalition structure 10
A partition of the set of agents into disjoint coalitions.

coarse correlated equilibrium (CCE) 13
A joint distribution over actions such that no player gains by deviating to a fixed action before seeing the recommendation, which is the outcome of no-regret learning.

coefficient of optimism 1
The weight in Hurwicz's criterion that balances the best and the worst outcomes of an action.

cold start 6
The problem a recommender has when it has no data about a new user or item.

collective decision 12
A decision made by a group of individuals through the aggregation of their preferences or judgments.

collusion 4
Secret coordination between participants to manipulate an outcome at the expense of others.

combination (binomial coefficient) 0
The number of ways to choose k items from n without regard to order, written n choose k.

combinatorial auction 14
An auction in which bidders may bid on bundles of items, with values that need not be additive.

combinatorial game 2
A game of perfect information with no chance moves in which players alternate moves, such as Nim or chess.

commitment 14
The ability of a player to bind themselves to an action before others move, giving a strategic advantage.

common knowledge 2
Information that all players know, know that everyone knows, and so on without end.

common prior 4
The assumption that all players' beliefs about types are derived from the same shared prior distribution.

common value 5
A setting in which the item has the same value to everyone, though bidders may hold different estimates of it.

competitive ratio 9
The worst-case ratio between the performance of an online algorithm and that of an optimal offline algorithm.

complement (of an event) 0
The set of all outcomes of an experiment that are not in the event.

complementary goods 5
Goods whose value to a buyer is higher when obtained together, such as left and right shoes.

complete graph 8
A graph in which every pair of distinct vertices is joined by an edge.

complete information 2
A game in which all players know the structure of the game, including all players' payoffs and actions.

completeness / continuity / independence axioms 1
The set of conditions on a preference relation that guarantee it can be represented by expected utility.

completion time 9
The time at which a job or flow finishes, used as a performance measure in scheduling and routing.

computational social choice 7
The field that studies the computational aspects of aggregating preferences, such as the complexity of voting rules.

concave 1
Describing a function whose line segment between any two points lies on or below its graph; used for risk-averse utilities.

conditional cooperator 10
A player who cooperates if a sufficient share of the others also cooperate.

conditional probability 0
The probability of an event given that another event is known to have occurred.

Condorcet jury theorem 12
The theorem that a group of independent voters with accuracy above one half gets more accurate as the group grows.

Condorcet loser 7
A candidate who loses every pairwise contest against all other candidates.

Condorcet paradox / cycle 7
The phenomenon in which majority preferences form a cycle, so that no Condorcet winner exists.

Condorcet winner 7
A candidate who beats every other candidate in pairwise majority contests.

confidence interval 14
A range of values computed from data that contains the true parameter with a stated probability.

confusion matrix 11
A table counting a classifier's true and false positives and negatives.

congestion game 8
A game in which each player chooses resources and the cost of a resource depends on the number of players using it.

congestion pricing 9
A pricing scheme that charges users for the congestion they cause so that equilibrium traffic approaches the optimum.

congestion toll 9
A fee charged for using a road or edge, designed to change drivers' incentives.

consensus 12
Agreement among distributed participants on a single value or state.

constant-product pool 14
A liquidity pool in which the product of the reserve quantities of the two assets is kept constant.

constant-sum game 2
A game in which the sum of all players' payoffs is the same in every outcome.

Contract Net Protocol 10
A task-allocation protocol in which a manager announces a task, agents bid, and the manager awards the contract.

convention / social law 10
A rule or norm adopted by agents to coordinate behavior and avoid conflict.

convergence 13
The property that a learning process or dynamic approaches a limiting strategy or value over time.

convex 1
Describing a function whose line segment between any two points lies on or above its graph; used for risk-seeking utilities.

convex game 10
A cooperative game in which the benefit of joining a coalition grows with the size of the coalition, so its core is non-empty.

cooperate (Prisoner's Dilemma action) 13
The action in the Prisoner's Dilemma that benefits both players if both choose it but can be exploited.

cooperative game 2
A game in which groups of players can form binding agreements and the focus is on how to divide the joint gains.

coordination failure 3
A situation in which players end up at a worse outcome because they cannot coordinate on a better equilibrium.

coordination game 3
A game in which players are better off when they choose the same or compatible actions.

coordination ratio 9
The term used by Koutsoupias and Papadimitriou for the ratio of the worst equilibrium cost to the optimum, now called the price of anarchy.

Copeland rule 7
A voting rule that scores each candidate by pairwise wins minus pairwise losses.

core 10
The set of allocations of a cooperative game's total value that no coalition can improve upon by acting on its own.

correlated equilibrium (CE) 3
A distribution over joint actions, given by a mediator, such that no player wants to deviate from the recommended action.

counterfactual regret minimization (CFR) 13
An algorithm for solving imperfect-information games by minimizing counterfactual regret at each information set.

Cournot duopoly 3
A model of competition in which firms simultaneously choose quantities and the market price depends on the total.

critical (swing) member 10
A member whose removal turns a winning coalition into a losing one.

critical bid 4
The lowest bid at which a bidder still wins, which determines their payment in a truthful auction.

critical mass 6
The minimum number of users needed for a platform to become self-sustaining.

cross-side (indirect) network effect 6
An effect in a two-sided market in which more users on one side make the platform more valuable to the other side.

cross-validation 14
A method for estimating how well a model will generalize by training and testing on different subsets of the data.

cut query 11
A query in a cake-cutting protocol that asks an agent to cut the cake at a point where a given value is reached.

cut-and-choose 11
A fair-division protocol in which one agent divides a resource into parts and the other chooses a part.

cutoff test 2
A rule that stops a game-tree search when a condition, such as reaching a depth limit, is met.

D

D'Hondt method 7
A proportional-representation method that allocates seats by successively dividing each party's votes by 1, 2, 3, and so on.

data drift 11
A change over time in the distribution of the data a model sees, which can degrade its performance.

decentralized exchange (DEX) 14
A trading venue run by smart contracts rather than a central intermediary.

decision making 1
The process of choosing among alternatives in order to attain a goal.

decision node 2
A node in a game or decision tree at which a player chooses an action.

decision tree 1
A tree that represents a sequence of decisions, chance events, and their outcomes.

decision under risk 12
A decision in which the probabilities of the outcomes are known.

decision under uncertainty 1
A decision in which the probabilities of outcomes are not known.

deep Q-network (DQN) 13
A deep-learning method that approximates the action-value function of Q-learning with a neural network.

defect (Prisoner's Dilemma action) 2
The action in the Prisoner's Dilemma that is individually best regardless of what the other player does.

degenerate game 3
A game that violates the usual genericity conditions, for example with ties in payoffs, and may have infinitely many equilibria.

deliberative agent 10
An agent that reasons about its goals and plans before acting.

demographic parity 11
A fairness criterion requiring that the rate of positive predictions be the same across groups.

depth-limited search 2
A game-tree search that explores only to a fixed depth and then applies an evaluation function.

derivative 0
The rate at which a function changes with respect to its input.

dictator game 12
A game in which one player proposes how to divide a sum and the other must accept the division.

differential privacy 14
A formal privacy guarantee under which the output of an analysis barely changes if any one individual's data are changed.

Digital Markets Act (DMA) 6
A European Union regulation that sets obligations for large online platforms designated as gatekeepers.

Dijkstra's algorithm 8
An algorithm that finds shortest paths from a source in a graph with non-negative edge weights.

diminishing marginal utility 1
The principle that each additional unit of a good adds less utility than the one before.

direct mechanism 4
A mechanism in which each agent reports their type directly.

direct network effect 6
An effect in which the value of a product to a user increases with the number of other users of the same kind.

discount factor 0
A factor between 0 and 1 by which future payoffs are multiplied to reflect their lower present value.

discriminant 0
The expression b squared minus 4ac that determines the number of real roots of a quadratic equation.

discriminatory (pay-as-bid) auction 5
An auction in which winners pay what they bid, with multiple units sold at their own bid prices.

diversity prediction theorem 12
The theorem that the collective error of a crowd equals the average individual error minus the diversity of the predictions.

doctrinal paradox (discursive dilemma) 7
A situation in which majority voting on individual propositions yields a collective position that is logically inconsistent.

Dodgson rule 7
A voting rule that elects the candidate who needs the fewest swaps of adjacent candidates in voters' rankings to become a Condorcet winner.

dominance by a mixed strategy 3
A situation in which a pure strategy is dominated by a randomization over other strategies, though not by any single one.

dominant strategy 2
A strategy that is at least as good as any other regardless of what the other players do.

dominant-strategy equilibrium 3
An outcome in which every player plays a dominant strategy.

dominant-strategy incentive compatible (DSIC) 4
A property of a mechanism in which reporting one's true type is a dominant strategy.

dominated strategy 3
A strategy that is worse than another strategy no matter what the opponents do.

double auction 5
A market in which buyers and sellers simultaneously submit bids and asks.

dummy (null) player 10
A player who adds nothing to any coalition they join.

Dutch (descending) auction 5
An auction in which the price starts high and falls until a bidder accepts it.

Duverger's law 7
The observation that plurality voting in single-seat districts tends to produce two-party systems.

E

ECMP (equal-cost multi-path) 8
A routing technique that splits traffic among several paths of equal cost.

edge 8
A connection between two nodes of a graph.

efficiency (allocative / social) 4
The property that an outcome maximizes the total value or welfare of the participants.

efficiency axiom 10
The axiom that a solution concept must distribute the entire value of the grand coalition.

efficiency loss 9
The reduction in total welfare caused by selfish behavior compared with an optimum.

efficient 4
Describing an outcome or allocation from which no change can improve total welfare.

egalitarian (Rawlsian) welfare 11
A social welfare measure that evaluates an outcome by the utility of the worst-off individual.

egalitarian criterion 12
A decision criterion that maximizes the utility of the least well-off member.

EIP-1559 14
An Ethereum proposal that introduced a base fee burned for each transaction and a tip for the block producer.

Ellsberg paradox 1
The paradox showing that people prefer bets with known probabilities to bets with unknown probabilities, contradicting subjective expected utility.

emergent behavior 10
Behavior of a system that arises from the interaction of its parts and is not specified in any one of them.

empirical revenue 14
The revenue that a mechanism earns on observed data, such as past bids.

endowment effect 12
The tendency to value an item more highly simply because one owns it.

English (ascending) auction 5
An auction in which the price rises until only one bidder remains.

entry (deterrence) game 2
A sequential game in which a potential entrant decides whether to enter a market and the incumbent chooses whether to fight.

envy-free equilibrium (GSP) 6
An equilibrium of the generalized second-price auction in which no bidder envies the slot of another at current prices.

envy-free up to one good (EF1) 11
A fairness criterion under which any envy can be removed by dropping a single good from the envied agent's bundle.

envy-freeness 11
The property that every agent prefers their own allocation to that of any other agent.

episode (RL) 13
One complete run of an agent interacting with an environment from a start state to a terminal state.

equal opportunity 11
A fairness criterion requiring equal true positive rates across groups.

equal split 10
A division in which each player of a cooperative game receives the same share of the total value.

equalized odds 11
A fairness criterion requiring equal true positive and false positive rates across groups.

equilibrium selection 3
The problem of deciding which of several equilibria players will actually play.

evaluation function 2
A function that estimates the value of a non-terminal game position for a player.

evidence 1
The data or observations used to update a belief.

evolutionarily stable strategy (ESS) 13
A strategy that, once adopted by a population, cannot be invaded by a small group of mutants.

EXP3 (adversarial bandit) 13
A bandit algorithm with regret guarantees against an adversarial sequence of rewards, using exponential weights.

expected loss 1
The average loss, or regret, an action is expected to incur over the possible states of nature.

expected payoff 2
The average payoff a player obtains, weighted by the probabilities of the outcomes.

expected utility 1
The probability-weighted average of the utilities of a lottery's outcomes.

expected value 0
The probability-weighted average of the values a random variable can take.

expected value of perfect information (EVPI) 1
The most a decision maker would pay for complete information about the state before deciding.

expected value of sample information (EVSI) 1
The most a decision maker would pay for a sample of imperfect information before deciding.

expert (online learning) 13
A source of advice whose predictions a learner follows or compares against in online learning.

explainability 11
The ability to explain in human-understandable terms why an algorithm produced a given output.

exploration–exploitation 13
The trade-off between trying new actions to gain information and choosing the best-known action to gain reward.

exponential mechanism 14
A differentially private mechanism that selects an output with probability proportional to the exponential of its utility.

extensive form 2
A representation of a game as a tree that shows the order of moves, information, and payoffs.

external regret 13
The difference between the total reward of the best fixed action and that of the algorithm over time.

externality 4
A cost or benefit that an agent's action imposes on others without payment.

F

facility location 4
The problem of placing one or more facilities to serve agents located on a line or network.

factorial 0
The product of all positive integers up to n, written n!, which counts the orderings of n items.

fair cost sharing 9
A network-design game in which the cost of each edge is divided equally among the players who use it.

fair division 11
The problem of dividing resources among agents so that the outcome is considered fair.

fairness 11
The principle that outcomes or decisions should treat individuals and groups equitably under some formal definition.

false negative rate (FNR) 11
The share of actual positives that a classifier wrongly labels negative.

false positive rate (FPR) 11
The share of actual negatives that a classifier wrongly labels positive.

false-name (sybil) bidding 4
Manipulation in which one participant takes part under several fake identities to gain an advantage.

federated learning 10
A machine-learning approach in which several parties train a shared model without sharing their raw data.

fee burning 14
Permanently destroying part of the transaction fees instead of paying them to the block producer.

feedback loop 6
A process in which a system's outputs influence its future inputs, which can amplify or damp an effect.

fictitious play 13
A learning process in which each player best-responds to the empirical frequency of the opponents' past actions.

finite / infinite horizon 2
Whether a game or decision problem lasts for a limited number of periods or continues indefinitely.

finite game 2
A game with a finite number of players, each with a finite set of strategies.

finite improvement property 8
The property that every sequence of improving unilateral moves terminates, which guarantees a pure equilibrium.

first-order condition 5
A necessary condition for an optimum that sets the first derivative to zero.

first-price sealed-bid auction 5
An auction in which bidders submit sealed bids and the highest bidder wins and pays their own bid.

fitness 13
A measure of an individual's or strategy's reproductive success in evolutionary models.

fixed-point theorem (Brouwer, Kakutani) 3
A theorem guaranteeing that certain continuous maps have a point that they leave unchanged; it is used to prove the existence of equilibria.

FIXP 3
The complexity class of problems that are equivalent to computing a fixed point of a continuous function.

focal point (Schelling) 3
An outcome that players choose because it stands out as natural, even without communication.

folk theorem 13
The theorem that, in infinitely repeated games, any feasible payoff above the minimax level can be sustained as an equilibrium.

follow the leader 13
A strategy that plays the action that performed best in the past.

four-fifths rule (disparate impact) 11
A rule of thumb under which a selection rate for a group below 80 percent of the highest rate is evidence of adverse impact.

framing effect 12
The finding that people's decisions change depending on how the same options are presented.

Free Money game 2
A game used in the course in which players can claim a prize, illustrating how incentives shape cooperation.

free rider 10
An agent who benefits from a shared resource or public good without paying for it.

free riding 10
Enjoying the benefits of a collective good while contributing little or nothing to it.

front-running 14
Acting on advance knowledge of a pending transaction to profit by placing one's own transaction ahead of it.

G

Gale–Shapley algorithm (deferred acceptance) 4
An algorithm that finds a stable matching between two sides by having one side propose and the other hold the best offer so far.

game 2
A situation in which players make interdependent decisions and each outcome gives every player a payoff.

game of Chicken 2
A game in which two players each prefer to continue but the worst outcome occurs if both do.

game theory 2
The mathematical study of strategic interaction among rational decision makers.

game tree 2
A tree whose nodes are game positions and whose branches are moves.

gaming (a metric) 11
Optimizing a measure to look good on it without improving what the measure was meant to capture.

gatekeeper (DMA) 6
A large online platform designated under the Digital Markets Act as a critical intermediary between businesses and users.

generalization 14
The ability of a model or mechanism to perform well on data it has not seen before.

Generalized Second Price (GSP) auction 6
The auction used for sponsored search in which each winner pays the bid of the bidder ranked just below.

geometric series 0
A sum of terms in which each term is a constant multiple of the previous one.

Gibbard–Satterthwaite theorem 7
The theorem that any non-dictatorial voting rule over three or more alternatives that can elect every alternative is manipulable.

glove game 10
A cooperative game in which left-glove and right-glove owners form pairs, illustrating the effect of scarcity on the core.

Goodhart's law 11
The principle that when a measure becomes a target, it ceases to be a good measure.

grand coalition 10
The coalition that contains all players of a cooperative game.

graph 8
A structure of nodes connected by edges, used to model networks.

greedy (algorithm) 10
An algorithm that makes the locally best choice at each step without reconsidering earlier choices.

greedy list scheduling 9
A scheduling algorithm that assigns each job to the currently least loaded machine.

grim trigger strategy 2
A strategy in a repeated game that cooperates until the opponent defects once and then defects forever.

group fairness 11
The requirement that outcomes be equitable across demographic groups rather than for each individual.

guess 2/3 of the average 2
A game in which players choose numbers and the winner is the one closest to two thirds of the average, illustrating iterated reasoning.

H

harmonic number H_n 9
The sum of the reciprocals of the first n integers, which grows like the natural logarithm of n.

Harsanyi transformation 4
A method that turns a game of incomplete information into a game of imperfect information by adding a chance move that selects the players' types.

Hawk–Dove game 13
A game of conflict in which two players each choose between an aggressive and a passive action, and the worst outcome occurs if both are aggressive.

heuristics 12
Simple rules of thumb that people use to make decisions quickly, at the cost of occasional systematic errors.

horizon effect 2
The distortion in a depth-limited search that arises because damaging events just beyond the search horizon are not seen.

housing market (house exchange) 4
A market in which each agent owns one house and agents trade houses without using money.

Hungarian algorithm 10
An algorithm that finds a minimum-cost assignment in a bipartite graph in polynomial time.

Hurwicz criterion 1
A decision rule that weights the best and the worst outcome of each action by a coefficient of optimism.

hybrid architecture 10
An agent architecture that combines reactive and deliberative layers.

I

impartial game 2
A game in which both players have the same moves available from any position, differing only in who moves.

imperfect information 2
A game in which players do not always know the earlier moves of the others or the current state.

implementation theory 4
The study of whether a given social goal can be achieved as the equilibrium outcome of some mechanism.

impossibility (incompatibility) theorems 11
Results showing that certain desirable properties cannot all be satisfied at the same time.

imputation 10
An allocation that gives each player at least their individual value and distributes the total value of the grand coalition.

incentive 4
A reward or penalty that influences an agent's choice of action.

incentive compatibility (IC) 4
The property that agents have no reason to misreport their private information.

incentive compatible 4
Describing a mechanism in which truthful reporting is the best strategy for each agent.

incomplete information 4
A game in which some players lack information about the types, payoffs, or actions of others.

incremental mean 13
A running average updated one observation at a time without storing all earlier values.

independence of irrelevant alternatives (IIA) 7
The requirement that the social ranking of two alternatives depends only on how voters rank those two.

independent cascade model 14
A model of diffusion in which each newly active node has a single chance to activate each of its neighbors with some probability.

independent private values (IPV) 5
An auction setting in which each bidder's value is private and independent of the others' values.

indifference principle 3
The principle that in a mixed equilibrium each player chooses probabilities that make the opponent indifferent between their strategies.

indirect mechanism 4
A mechanism in which agents take actions or send messages other than a direct report of their type.

individual fairness 11
The requirement that similar individuals be treated similarly.

individual rationality (IR) 4
The property that participation never leaves an agent worse off than not participating.

indivisible goods 11
Goods that cannot be divided among agents, such as a house or a car.

inequity aversion 12
The tendency to dislike outcomes in which one's payoff differs from those of others, in either direction.

influence maximization 14
The problem of choosing a small set of seed nodes so as to maximize the spread of influence in a network.

information asymmetry 6
A situation in which one party in a transaction has more or better information than the other.

information cascade 12
A situation in which people ignore their own private information and follow the observed actions of those before them.

information design (Bayesian persuasion) 14
The study of how a sender should choose the information structure to influence a receiver's beliefs and actions.

information set 2
A set of decision nodes among which a player cannot distinguish.

information structure 10
A specification of who knows what in a game, including the signals available to each player.

inspection game 3
A game in which an inspector chooses whether to check and an inspectee chooses whether to comply or cheat.

instant-runoff voting (IRV) 7
A voting system in which voters rank candidates and the candidate with the fewest first-place votes is eliminated repeatedly.

integral 0
The area under a curve; the limit of sums of function values over small intervals.

interchangeability (of equilibria) 3
The property of zero-sum games that if two equilibrium strategies exist, mixing them in any combination yields another equilibrium.

interdependent values 5
An auction setting in which each bidder's value depends on the information held by other bidders.

internal (swap) regret 13
Regret measured against modifying each occurrence of one action to another action, which leads to correlated equilibrium.

Internet service provider (ISP) 8
A company that provides access to the Internet and carries traffic for its customers.

iterated elimination of strictly dominated strategies (IESDS) 3
The procedure of repeatedly deleting strategies that are strictly dominated until no more can be removed.

iterative deepening 2
A search that runs depth-limited searches with increasing depth limits until time runs out.

J

Jensen's inequality 1
The inequality that the expected value of a convex function is at least the function of the expected value.

judgment aggregation 7
The study of how to combine individuals' judgments on logically connected propositions into a collective judgment.

K

k-approval 7
A voting rule in which each voter approves of exactly k candidates.

Kemeny rule 7
A rule that elects the ranking with the smallest total number of pairwise disagreements with the voters' rankings.

kidney exchange 4
A market in which patients with incompatible donors swap donors so that each receives a compatible kidney.

L

Laplace criterion 1
A decision rule that treats all states of nature as equally likely and picks the action with the best average.

large language model (LLM) 14
A neural network trained on large text corpora to generate and understand natural language.

largest remainder (Hare) method 7
A proportional-representation method that gives each party the whole number of quotas it earns, then allocates leftover seats by largest remainder.

last diminisher 11
A cake-cutting protocol for several agents in which each trims the piece in turn and the last one to trim receives it.

last iterate 13
The final strategy that a learning algorithm outputs, as opposed to the average of its past strategies.

latency (delay) function 8
A function giving the delay on an edge as a function of the flow it carries.

leaf (terminal node) 2
A node of a game tree with no children, at which the game ends.

learning architecture 10
The structure through which an agent improves its behavior from experience.

learning rate 13
A parameter that controls how large a step a learning algorithm takes in response to new information.

Lemke–Howson algorithm 3
An algorithm that finds a Nash equilibrium of a two-player game by pivoting along the edges of a polytope.

level-k reasoning 2
A model in which a level-0 player acts naively and a level-k player best-responds to a level-(k minus 1) player.

likelihood 1
The probability of the observed evidence under a given hypothesis.

linear programming (LP) 3
The optimization of a linear objective subject to linear constraints.

link 8
A connection between two nodes in a network, with a capacity and a cost or delay.

Lipschitz condition 11
A bound stating that the change in a function is at most a constant times the change in its input.

load balancing 9
The problem of distributing jobs or traffic across machines or paths to even out loads.

locally envy-free equilibrium (GSP) 6
An equilibrium of the generalized second-price auction in which no bidder wants to swap with the slot directly above.

logit choice (logit response) 12
A stochastic choice rule in which actions with higher payoffs are chosen with exponentially higher probability.

logit function 0
The function that maps a probability to its log-odds, the logarithm of p divided by one minus p.

loss aversion 12
The tendency to feel losses more strongly than equal-sized gains.

loss function 11
A function that quantifies the cost of a model's prediction error, which training seeks to minimize.

lottery 1
A probability distribution over outcomes, used to represent a risky choice.

M

M/M/1 queue 8
A queueing model with Poisson arrivals, exponential service times, and one server.

majority game 10
A game in which a coalition wins if it contains more than half of the players.

majority rule 7
A rule that picks the alternative preferred by more than half of the voters.

makespan 9
The completion time of the last job to finish, a standard objective in scheduling.

manager / contractor (contract net) 10
The roles in the Contract Net Protocol: the manager announces tasks and contractors bid and perform them.

manipulation-resistant 5
Describing a mechanism or rule that makes it hard or impossible for participants to gain by misreporting.

marginal contribution 10
The increase in a coalition's value from adding a given player.

marginal cost 8
The extra cost of producing or serving one additional unit.

marginal per-capita return (MPCR) 10
The return that each contributor receives from one unit contributed to a public good, in public goods experiments.

marginal-cost toll 9
A toll equal to the marginal cost that a user imposes on the others, which induces the socially optimal flow.

market 5
A place or mechanism where buyers and sellers exchange goods and services.

market design 5
The design of rules and institutions for markets so that they work efficiently.

market for lemons 6
The market in which sellers know product quality but buyers do not, so that low-quality goods drive out high-quality ones.

market maker 5
A participant who continuously quotes both buy and sell prices, providing liquidity.

marketplace 6
A platform where multiple sellers offer goods or services to buyers.

markup 6
An amount added to cost to reach the selling price.

matcher / mismatcher (Matching Pennies roles) 2
The two roles in Matching Pennies: one player wants the coins to match and the other wants them to differ.

matching market 4
A market in which participants are matched in pairs, with preferences on both sides and often no prices.

Matching Pennies 2
A zero-sum game in which one player wins if both coins match and the other wins if they differ.

max-min fairness 8
An allocation in which the smallest share is as large as possible, and then the next smallest, and so on.

maximal extractable value (MEV) 14
The profit that a block producer or others can obtain by reordering, inserting, or censoring transactions in a block.

maximax 1
A decision rule that picks the action with the best possible outcome.

maximin 3
A decision rule that picks the action whose worst possible outcome is best.

maximin (Simpson) rule 7
A voting rule that elects the candidate whose worst pairwise defeat is smallest.

maximize 2
To choose the action or value that makes an objective function as large as possible.

May's theorem 7
The theorem that majority rule is the only two-alternative rule that is anonymous, neutral, and positively responsive.

McAfee double auction 5
A double auction that finds the breakeven trade size and is truthful and budget balanced while trading all but one possible trade.

mechanism 4
A set of rules that maps participants' reports to an outcome and payments.

mechanism design 4
The design of rules of a game so that self-interested players' behavior leads to a desired outcome.

mechanism design without money 4
Mechanism design in which payments are not allowed, so that incentives come only from the allocation rule.

median mechanism 4
A mechanism that selects the median of the reported peaks, which is truthful for single-peaked preferences.

median voter theorem 7
The theorem that under single-peaked preferences majority rule selects the median voter's most-preferred position.

mempool 14
The pool of pending transactions waiting to be included in a block.

Metcalfe's law 8
The claim that the value of a network grows in proportion to the square of the number of its users.

midpoint rule 5
A rule that sets the price halfway between the buyer's bid and the seller's ask.

minimax 3
A decision rule for zero-sum games in which each player minimizes the maximum the opponent can gain.

minimax regret (Savage) 1
A decision rule that picks the action whose maximum regret across states is smallest.

minimax theorem (von Neumann) 3
The theorem that in every finite two-player zero-sum game the maximin and minimax values are equal.

minimize 8
To choose the action or value that makes an objective function as small as possible.

mixed (strategy) equilibrium 3
An equilibrium in which at least one player randomizes over several pure strategies.

mixed Nash equilibrium 3
A Nash equilibrium in which players may choose probability distributions over their pure strategies.

mixed strategy 3
A probability distribution over a player's pure strategies.

mixed-motive 10
Describing a situation in which agents have both cooperative and competitive motives.

model card 11
A short document describing a machine-learning model's intended use, performance, and limitations.

monotone allocation rule 4
An allocation rule in which a bidder who raises their bid never receives less than before.

monotonicity 7
The property that improving a candidate's ranking cannot cause them to lose.

Monte Carlo tree search (MCTS) 2
A search algorithm that evaluates positions by averaging the results of many random simulated games.

move ordering 2
The ordering of moves examined in a search so that the best moves are tried first, increasing pruning.

moving knife (Dubins–Spanier) 11
A cake-cutting procedure in which a knife moves across the cake until some player calls out that the piece is worth a fair share.

multi-agent reinforcement learning 14
Learning in which several agents adapt their policies simultaneously in a shared environment.

multi-agent system (MAS) 10
A system composed of multiple interacting agents that pursue their own or shared goals.

multi-armed bandit 13
A problem in which a learner repeatedly chooses among options with unknown reward distributions to maximize cumulative reward.

multi-homing 6
The practice of using several platforms at once.

multi-unit auction 5
An auction that sells several identical units of a good.

multiple equilibria 3
A situation in which a game has more than one equilibrium.

multiplicative weights (Hedge) 13
An online learning algorithm that exponentially updates the weight of each action according to its past performance.

Myerson's lemma 4
A lemma characterizing truthful mechanisms in single-parameter settings: the allocation must be monotone and payments are determined by it.

Myerson–Satterthwaite theorem 4
The theorem that no mechanism for bilateral trade can be efficient, incentive compatible, individually rational, and budget balanced.

N

N-position 2
A game position from which the next player to move wins with correct play.

Nash equilibrium (NE) 3
A strategy profile in which no player can gain by deviating unilaterally.

Nash social welfare 11
A welfare measure equal to the product of the players' utilities, or the sum of their logarithms.

Nash's existence theorem 3
The theorem that every finite game has at least one Nash equilibrium, possibly in mixed strategies.

natural frequencies 1
A way of presenting probabilities as counts of cases out of a sample, which makes Bayesian reasoning easier.

natural logarithm 0
The logarithm with base e, the inverse of the exponential function.

Nature (chance player) 4
A pseudo-player that makes the random moves in a game.

negamax 2
A simplification of minimax for zero-sum games in which the same procedure is applied at every node by negating the child's value.

negative predictive value (NPV) 11
The share of negative predictions that are truly negative.

neighboring datasets 14
Two datasets that differ in the data of a single individual, used to define differential privacy.

network 8
A set of nodes connected by links.

network design 8
The problem of building or improving a network so as to minimize cost or maximize performance.

network effect 6
The phenomenon in which a product's value to a user depends on how many others use it.

network formation game 8
A game in which players choose which links to create, balancing the cost of links against the benefits of connectivity.

neutrality 7
The requirement that a voting rule treat all candidates the same, so that renaming candidates does not change the result.

nim-sum 2
The bitwise exclusive-or of the pile sizes in Nim; the position is losing for the player to move if and only if it is zero.

no-regret 13
A property of a learning algorithm whose average regret against the best fixed action vanishes over time.

node 8
A vertex of a graph.

noise (trembles) 13
Small random errors in players' actions or choices.

non-cooperative game 2
A game in which players cannot make binding agreements and each acts in their own interest.

non-credible threat 2
A threat that a player would not carry out if called upon, because doing so would hurt them.

non-dictatorship 7
The requirement that no single voter's preferences always determine the social outcome.

non-stationary 10
Describing an environment whose behavior changes over time, for example as other agents learn.

nonatomic (flow) 8
Describing a routing model in which traffic is infinitely divisible and each player is infinitesimally small.

normal form (strategic form) 2
A representation of a game as a matrix that lists the players' strategies and the resulting payoffs.

normal play convention 2
The convention that the player who cannot move loses.

NP-hard 3
Describing problems at least as hard as every problem in NP, which are believed not to be solvable in polynomial time.

nucleolus 10
The allocation in a cooperative game that lexicographically minimizes the largest dissatisfactions of coalitions.

O

obedience constraints 3
Conditions requiring that a player prefers to follow a recommendation, which define a correlated equilibrium.

oddness theorem (Wilson) 3
The theorem that almost every finite game has an odd number of Nash equilibria.

on-chain governance 14
A scheme in which the rules of a blockchain are changed by votes recorded on the blockchain itself.

one-shot game 3
A game that is played only once, with no repeated interaction.

online allocation 6
The problem of assigning resources to requests as they arrive, without knowledge of the future.

online platform 6
A digital service that connects distinct groups of users, such as buyers and sellers.

optimal auction (Myerson) 5
The auction that maximizes the seller's expected revenue given the distribution of bidders' values.

optimistic bias (in-sample overestimation) 14
The tendency of a model or mechanism to appear better on the data it was tuned on than on new data.

optimization 14
The process of finding the best solution among the feasible ones according to an objective.

order book 5
A record of all outstanding buy and sell orders for an asset, organized by price.

ordinal utility 2
A utility function that represents only the ranking of outcomes and not their intensity.

OSPF (Open Shortest Path First) 8
A link-state routing protocol that computes shortest paths using Dijkstra's algorithm.

outcome 2
The result of a play of a game, determined by the players' joint choices and chance.

overconfidence 12
The tendency to be more certain of one's beliefs or abilities than is justified.

overfitting 14
The situation in which a model fits the training data too closely and performs poorly on new data.

P

P-position 2
A game position from which the previous player to move wins with correct play.

pairwise majority graph 7
A graph whose nodes are candidates and whose edges show which candidate wins a pairwise majority contest.

pairwise stability 8
A condition on a network in which no pair of nodes would gain by forming a link and no node by deleting one.

Pareto (unanimity) condition 7
The requirement that if every voter prefers one alternative to another, then so does the social ranking.

Pareto dominance 3
The relation in which one outcome is at least as good as another for every player and strictly better for some.

Pareto efficiency 11
The state in which no one can be made better off without making someone else worse off.

Pareto optimal 2
Describing an outcome that no other outcome Pareto-dominates.

Pareto optimality 2
The property of an outcome from which no change can help one player without hurting another.

partial derivative 0
The derivative of a multivariable function with respect to one variable while the others are held fixed.

participation equilibrium 6
An equilibrium describing how many users or sellers join a platform.

payment rule 5
The part of a mechanism that determines how much each participant pays or receives.

payoff 2
The value that a player receives from an outcome of the game.

payoff function 2
A function that assigns to each outcome of a game the payoff for each player.

payoff matrix 2
A table that lists the payoffs for each combination of players' strategies.

peer prediction 14
A mechanism that rewards agents for reports that agree with those of peers, without knowing the true answer.

peering 8
An arrangement in which two networks exchange traffic directly, often without payment.

perfect information 2
A game in which every player knows the full history of play at each decision.

Pigou network 9
A two-edge network with one constant-cost edge and one load-dependent edge, used to show the inefficiency of selfish routing.

Pigouvian tax 12
A tax equal to the external cost an activity imposes, used to correct an externality.

pivotal player 10
A player whose decision determines whether a coalition wins or loses.

player 2
A decision maker in a game.

PLS-complete 8
The complexity class for local search problems, whose complete problems are believed to be hard.

plurality rule 7
A rule in which the candidate with the most first-place votes wins.

ply 2
One move by one player in a game tree; a half-move.

policy (agent / RL) 13
A rule that tells an agent which action to take in each state.

positional scoring rule 7
A voting rule that gives points to candidates according to their position in each ranking and elects the highest total.

positive predictive value (PPV), precision 11
The share of positive predictions that are truly positive.

positive responsiveness 7
The requirement that if a winner improves in some ballots, the winner remains the unique winner.

posted price 14
A fixed price offered to buyers on a take-it-or-leave-it basis.

posterior 0
The updated probability of a hypothesis after taking evidence into account.

potential function 8
A function whose change under one player's unilateral deviation equals the change in that player's cost.

potential game 8
A game that admits a potential function, which guarantees a pure Nash equilibrium.

power index 10
A numerical measure of a player's influence in a cooperative or voting game.

PPAD-complete 3
The complexity class of problems that are equivalent to finding a Nash equilibrium of a two-player game.

pre-processing / post-processing 11
Methods that reduce bias in data before training or in a model's outputs afterward.

prediction market 14
A market in which participants trade contracts whose payoffs depend on the outcome of an event, so prices reflect aggregate beliefs.

predictive parity 11
A fairness criterion requiring equal positive predictive value across groups.

preference elicitation 7
The process of learning agents' preferences through queries.

preference profile 7
A list that gives each voter's ranking of all alternatives.

price discovery 5
The process by which a market arrives at prices reflecting supply and demand.

price impact 14
The change in an asset's price caused by a trade.

Price of Anarchy (PoA) 9
The ratio between the cost of the worst Nash equilibrium and the cost of the social optimum.

price of fairness 11
The ratio between the optimal welfare and the welfare of the best fair outcome, measuring what fairness costs.

Price of Stability (PoS) 9
The ratio between the cost of the best Nash equilibrium and the cost of the social optimum.

price structure 6
How the total price of a service is split between the two sides of a platform.

price-matching guarantee 14
A promise to match or beat a competitor's lower price, which can facilitate tacit collusion.

pricing 6
The setting of the amounts that buyers pay for goods or services.

principal–agent problem 14
A situation in which one party acts on behalf of another whose interests differ and whose actions they cannot fully observe.

prior 0
The probability distribution of a quantity before any evidence is taken into account.

priority fee (tip) 14
An optional payment added to a transaction to induce the block producer to include it sooner.

Prisoner's Dilemma 2
A two-player game in which each player's dominant strategy is to defect, yet both would be better off if both cooperated.

privacy 11
The right of individuals to control information about themselves.

private information 4
Knowledge about a player's own type or preferences that others do not have.

private value 5
A setting in which each bidder's value for the item is known only to the bidder.

pro-activeness 10
The ability of an agent to take the initiative and pursue goals rather than only respond to events.

probability density 0
A function whose integral over an interval gives the probability that a continuous random variable falls in it.

probability distribution 0
A description of the probabilities with which a random variable takes each of its possible values.

probability vector 0
A list of non-negative numbers that sum to one, representing a distribution over a finite set.

probability weighting 12
The tendency to overweight small probabilities and underweight large ones when evaluating prospects.

product rule (counting) 0
The counting rule that if one task can be done in m ways and another in n ways, the two can be done in m times n ways.

profitable deviation 6
A change of strategy by one player that raises their own payoff.

prompt 14
The input text given to a language model to elicit a response.

prompt injection 14
An attack in which text supplied to a language model overrides or alters its intended instructions.

proof of stake 14
A blockchain consensus mechanism in which validators are chosen to create blocks in proportion to the assets they stake.

proportional fairness 8
An allocation in which no change can raise the total of the relative changes in individual shares above zero.

proportional representation 7
An electoral system in which the share of seats a party gets matches the share of votes it receives.

proportionality 11
The principle that each agent receives at least their fair share of the total value.

proposing side 4
The side of a matching market whose members make the offers in the deferred acceptance algorithm.

prospect theory 12
A descriptive theory of decision making under risk that uses a reference point, loss aversion, and probability weighting.

protected attribute 11
A characteristic such as race, sex, or religion on which discrimination is prohibited.

proxy variable 11
A feature that is correlated with a protected attribute and can lead to indirect discrimination.

public good 10
A good that people can use without reducing its availability to others and from which no one can be excluded.

public goods game 10
A game in which players decide how much to contribute to a common pool whose total is multiplied and shared equally.

pure coordination game 2
A game in which players' interests are fully aligned and they only need to choose the same action.

pure strategy 2
A strategy in which a player chooses a single action with certainty.

pure(-strategy) Nash equilibrium 3
A Nash equilibrium in which every player plays a pure strategy.

Q

Q-learning 13
A reinforcement-learning algorithm that learns the value of each state–action pair without a model of the environment.

qualified majority 7
A voting rule requiring more than a simple majority, such as two thirds, for a proposal to pass.

quality score 6
An estimate of how relevant and good an ad is, used with the bid to rank ads in sponsored search.

quantal response equilibrium 12
An equilibrium concept in which players choose better actions with higher probability but make errors.

quasi-linear utility 4
A utility function that is linear in money, so that utility equals value minus payment.

quiescence search 2
An extension of a depth-limited search that continues searching unstable positions until they are quiet.

quota 10
The number of votes or weight required for a coalition to win in a weighted voting game.

R

random seed 14
The initial value used to start a pseudo-random number generator, which makes experiments reproducible.

random serial dictatorship (RSD) 4
A mechanism in which agents in a random order each pick their favorite item among those remaining.

random variable 0
A function that assigns a numerical value to each outcome of a random experiment.

rank aggregation 7
The problem of combining several rankings of alternatives into a single consensus ranking.

rational player 2
A player who chooses actions that maximize their expected utility given their beliefs.

rationality 2
The assumption that players choose the actions that best serve their preferences.

re-auction (re-tendering) 10
A procurement process in which a contract is put out to bid again after the first round.

reactive architecture 10
An agent architecture that responds directly to the environment through condition–action rules, without internal models.

reactivity 10
The ability of an agent to perceive its environment and respond to changes in a timely way.

reciprocity 12
The tendency to respond to kind actions with kindness and to unkind actions with unkindness.

reference point 12
The point against which outcomes are judged as gains or losses in prospect theory.

regret 13
The difference between the payoff of the best fixed action in hindsight and the payoff the algorithm actually obtained.

regret matching 13
A no-regret algorithm that chooses actions with probability proportional to their accumulated positive regret.

regret minimization 13
The goal of making an algorithm's regret against the best action in hindsight small over time.

regular distribution 4
A distribution whose virtual valuation function is monotone, so that the optimal auction takes a simple form.

reinforcement learning 13
Learning in which an agent learns to act by interacting with an environment and receiving rewards.

related machines 9
A scheduling model in which machines have different speeds, so that the time to run a job is its size divided by the machine's speed.

repeated game 13
A game in which the same stage game is played several times by the same players.

replicator dynamics 13
A dynamic in which the share of each strategy in a population grows in proportion to how much it outperforms the average.

report 4
The message a participant sends to a mechanism, which may or may not be truthful.

reproducibility 14
The ability for others to obtain the same results by repeating a study or experiment.

reputation system 6
A system that collects and publishes feedback about participants to build trust.

reserve price 5
The minimum price below which the seller will not sell.

resource allocation 10
The problem of dividing limited resources among competing demands.

revelation principle 4
The principle that any outcome that can be implemented by some mechanism can also be implemented by a truthful direct mechanism.

revenue 5
The total payments a seller receives from an auction.

revenue equivalence theorem 5
The theorem that standard auctions with the same allocation and the same payment for the lowest type give the seller the same expected revenue.

revenue maximization 5
The goal of designing an auction to maximize the seller's expected revenue.

reverse auction 10
An auction in which sellers compete to offer the lowest price to a single buyer.

RIP (Routing Information Protocol) 8
A distance-vector routing protocol that uses hop count as its metric.

risk attitude 1
A person's tendency to avoid risk, to seek it, or to be indifferent to it.

risk dominance 3
A criterion that selects the equilibrium that is less costly to be wrong about.

risk premium 1
The amount by which the expected value of a lottery exceeds its certainty equivalent.

risk-averse 1
Describing a decision maker who prefers the expected value of a lottery to the lottery itself.

risk-loving / risk-seeking 1
Describing a decision maker who prefers a lottery to its expected value.

risk-neutral 1
Describing a decision maker who is indifferent between a lottery and its expected value.

Robertson–Webb query model 11
A model of fair division that allows only two kinds of queries: evaluate a piece and cut at a given value.

robust Price of Anarchy 9
A bound on the price of anarchy that also holds for coarse correlated equilibria and other outcomes of no-regret learning.

Rock–Paper–Scissors (RPS) 2
A zero-sum game in which each of two players picks rock, paper, or scissors and each choice beats one other and loses to one.

root (of a game tree) 2
The initial position from which the game starts in a game tree.

Rosenthal potential 8
The potential function of congestion games, equal to the sum over resources of the cumulative cost of loads.

round-robin 11
A procedure in which agents take turns choosing items in a fixed order.

router 8
A device that forwards data packets between networks.

routing 8
The selection of paths along which traffic travels through a network.

row / column player 2
The names for the two players in a two-player matrix game, whose strategies are rows and columns.

Rubinstein bargaining model 14
The model of alternating-offers bargaining with discounting, whose unique subgame perfect equilibrium gives an immediate agreement.

S

saddle point 3
A point that is a minimum in one direction and a maximum in another; in a zero-sum game, the pure-strategy equilibrium.

sample complexity 14
The number of samples a learning algorithm needs to reach a given accuracy with high probability.

sandwich attack 14
An attack in which an attacker places one transaction before and one after a victim's trade to profit from the price movement.

SARSA 13
An on-policy reinforcement-learning algorithm that updates values using the action actually taken next.

satisficing 12
Choosing an option that is good enough rather than searching for the best.

school choice 4
The problem of assigning students to schools based on their preferences and the schools' priorities.

sealed bid 5
A bid submitted in secret so that others cannot see it before the auction closes.

second derivative 0
The derivative of the derivative, which describes curvature.

second-order free riding 12
The problem of deciding who will bear the cost of punishing those who do not cooperate.

second-price (Vickrey) auction 5
An auction in which the highest bidder wins and pays the second-highest bid.

security game 3
A game in which a defender allocates limited resources to protect targets against an attacker.

seeding 6
The practice of giving initial users or content a boost so that a platform can start up.

selection rate 11
The proportion of applicants from a group that receive a positive decision.

self-fulfilling expectation 6
A belief that comes true because people act on it.

self-interest 2
The assumption that each agent pursues its own interests.

self-play 13
A training method in which an agent learns by playing against copies of itself.

self-preferencing 6
The practice by which a platform favors its own products or services over those of competitors.

selfish routing 8
Routing in which each user chooses the path that is best for themselves rather than for the whole system.

sensitivity (true positive rate) 1
The share of actual positives that a test or classifier correctly identifies.

sequential game 2
A game in which players move one after another and later players observe earlier moves.

sequential pairwise voting 7
A voting procedure that compares alternatives in a fixed agenda order, with the winner of each comparison facing the next alternative.

sequential updating 1
The process of revising beliefs step by step as each new piece of evidence arrives.

serial dictatorship 4
A mechanism in which agents in a fixed order choose their favorite item among those remaining.

series-parallel network 8
A network that can be built from a single edge by repeated series and parallel compositions.

shadow of the future 13
The idea that the prospect of future interaction encourages cooperation today.

SHAP (SHapley Additive exPlanations) 10
A method that explains a model's predictions by assigning each feature a Shapley value.

Shapley value 10
A solution concept that assigns each player their average marginal contribution over all orders in which the coalition could form.

Shapley–Shubik index 10
A power index that counts how often a player is pivotal when voters join in random order.

simple majority 7
A rule under which a proposal passes if it receives more than half of the votes.

simultaneous-move game 2
A game in which all players choose their actions at the same time without seeing the others' choices.

single point of failure 8
A component whose failure stops the whole system from working.

single-parameter environment 4
A setting in which each agent's private information is a single number, such as their value per unit.

single-peaked preferences 7
Preferences over alternatives on a line such that each voter's utility falls as an alternative gets farther from their ideal point.

slashing 14
The penalty in proof-of-stake systems in which a validator loses part of its staked assets for misbehaving.

slippage tolerance 14
The maximum difference between expected and executed price that a trader accepts.

slot (ad position) 6
A position on a results page at which an ad can be displayed.

smart contract 14
A program stored on a blockchain that runs automatically when predefined conditions are met.

smoothness, (λ, μ)-smooth game 9
A condition for games bounding the price of anarchy by two parameters λ and μ.

social ability 10
The ability of an agent to interact with other agents and humans through a communication language.

social choice 7
The theory of how to combine individual preferences into a collective decision.

social choice function (SCF) 4
A function that maps the profile of voters' preferences to a single chosen alternative.

social cost 8
The total cost borne by all users of a system.

social optimum 9
The outcome that minimizes total cost or maximizes total welfare, regardless of incentives.

social welfare 4
The sum of the utilities of all participants for an outcome.

social welfare function (SWF) 7
A function that maps the profile of voters' preferences to a collective ranking of alternatives.

softmax choice 13
A rule that picks each action with probability proportional to the exponential of its value divided by a temperature.

software-defined networking (SDN) 8
A network architecture that separates the control plane from the data plane so that it can be programmed centrally.

spectrum auction 5
An auction in which the government sells licenses for use of a range of radio frequencies.

sponsored search auction 6
An auction held for each search query, in which advertisers bid for ad slots on the results page.

Sprague–Grundy theorem 2
The theorem that every impartial game is equivalent to a Nim pile of some size, its Grundy value.

St. Petersburg paradox 1
A gamble with infinite expected value that people nevertheless value only at a small price, motivating utility theory.

stability 4
The property that no participant has an incentive to deviate from the outcome.

stable matching 4
A matching in which no two agents would both prefer each other to their assigned partners.

Stackelberg routing 8
A routing model in which a leader controls a share of the traffic and the remaining users respond selfishly.

Stag Hunt 2
A coordination game in which hunting a stag together gives the best payoff but hunting a hare is safe.

star / ring / tree / mesh topology 8
The common ways of arranging nodes and links in a network.

state–action value 13
The expected return of taking a given action in a given state and following a policy afterward.

stationary strategy 14
A strategy that depends only on the current state and not on the history.

statistical bias 14
A systematic difference between an estimate and the true value being estimated.

status quo bias 12
The preference for the current state of affairs over change.

strategic classification 11
The setting in which people alter the features a classifier uses in order to receive a better classification.

strategic interaction 2
A situation in which each participant's outcome depends on the choices of all participants.

strategic voting 7
Voting for something other than one's true preference in order to obtain a better result.

strategy 2
A complete plan that specifies what a player will do in every situation they may face.

strategy profile 3
A list that specifies one strategy for each player.

strategy set 3
The set of all strategies that are available to a player.

strategy-proof 4
Describing a mechanism in which telling the truth is a dominant strategy for every participant.

strategy-stealing argument 2
An argument showing that a second player cannot have a winning strategy, because the first player could copy it.

strict dominance 3
The relation in which one strategy gives a strictly higher payoff than another against every strategy of the opponents.

strict equilibrium 2
An equilibrium in which each player's strategy is the unique best response to the others'.

strong (exact) budget balance 4
The requirement that the payments in a mechanism sum exactly to zero.

subgame 2
The part of a game tree that starts at a single decision node and forms a game in its own right.

subgame perfect equilibrium (SPE) 2
A strategy profile that is a Nash equilibrium in every subgame.

submodularity 14
The property that the marginal benefit of adding an element decreases as the set grows.

subsidy 6
A payment from a platform or the government that lowers the price for one side of a market.

superadditive 10
Describing a characteristic function in which the value of two disjoint coalitions together is at least the sum of their separate values.

support (of a mixed strategy) 3
The set of pure strategies that a mixed strategy plays with positive probability.

support enumeration 3
A method of finding Nash equilibria by trying every pair of supports and solving the resulting equations.

surjectivity (onto) 7
The property that every alternative can be the outcome of the rule for some profile of preferences.

surplus (gains from trade) 5
The total benefit of a trade, equal to the buyer's value minus the seller's cost.

suspicious / generous Tit-for-Tat 13
Variants of Tit for Tat that begin with defection or occasionally forgive a defection.

switching cost 6
The cost that a user incurs when moving from one platform to another.

symmetric game 2
A game in which all players have the same strategies and the payoff depends only on the strategy chosen, not on who chose it.

symmetry axiom 10
The axiom that two players who contribute equally to every coalition receive the same value.

system optimum (Wardrop's 2nd principle) 8
The flow that minimizes total travel time over the network, regardless of individual incentives.

T

tactical voting (plurality) 7
Voting for a candidate other than one's favorite in a plurality election to avoid wasting one's vote.

tail probability 0
The probability that a random variable exceeds a given large value.

take-it-or-leave-it offer 4
An offer that the other party can only accept or reject, with no room for negotiation.

task allocation 10
The problem of deciding which agent performs which task in a multi-agent system.

task announcement / bid / award 10
The three stages of the Contract Net Protocol: announcing a task, receiving bids, and awarding the contract.

temperature (sampling) 14
A parameter that controls the randomness of a language model's output by scaling the probabilities.

Thompson sampling 13
A bandit algorithm that picks actions by sampling from the posterior probability that each is best.

threshold public good 12
A public good that is provided only if total contributions reach a threshold.

tie-breaking rule 7
A rule for choosing a winner when two or more alternatives are tied.

tight bound 9
A bound that matches the worst case exactly, so that it cannot be improved.

time average 13
The mean of a quantity computed over the whole sequence of rounds.

tipping 6
The point at which a platform's adoption accelerates and moves rapidly to dominance.

Tit-for-Tat (TFT) 13
A strategy for repeated games that cooperates in the first round and then copies the opponent's previous move.

to block (a coalition blocks) 10
To be able to improve a coalition's payoff by deviating from a proposed allocation.

token 14
A basic unit of text processed by a language model, or a digital asset on a blockchain.

Top Trading Cycles (TTC) 4
A mechanism for housing markets in which agents are traded along cycles of preferred houses, yielding a core allocation.

tournament 7
A complete directed graph in which each pair of vertices is joined by exactly one directed edge.

trade-off 4
A situation in which improving one desirable quality requires giving up another.

tragedy of the commons 12
The situation in which individuals acting in self-interest deplete a shared resource that would be better preserved collectively.

transaction fee mechanism 14
The rules by which a blockchain decides which transactions are included and what they pay.

transferable-utility (TU) game 10
A cooperative game in which the value of a coalition can be freely divided among its members.

transitivity 7
The property that if A is preferred to B and B to C, then A is preferred to C.

transparency 11
The principle that the workings and decisions of an algorithm be open to inspection.

transposition table 2
A table that stores the values of positions already searched so that they need not be searched again.

trigger strategy 2
A strategy in a repeated game that cooperates until a trigger event occurs and then switches to punishment.

true positive rate (TPR) 11
The share of actual positives that a classifier correctly labels positive.

trust game 12
A game in which one player may send money to another, which multiplies it, and the second may return some.

truthful bid(ding) 5
Bidding one's true value.

truthful reporting 4
Reporting one's private information honestly.

truthfulness 5
The property of a mechanism that participants do best by reporting their true information.

two-round (runoff) system 7
A voting system in which, if no candidate has a majority, the top two candidates face each other in a second round.

two-sided market 6
A market in which a platform serves two distinct groups that each benefit from the presence of the other.

type (private information of a player) 4
The private information of a player that determines their payoffs or preferences.

type profile 4
A list that specifies the type of each player.

U

Ultimatum game 2
A game in which a proposer offers a split and the responder may accept or reject it, in which case both get nothing.

uniform distribution 0
A distribution in which all values in a range are equally likely.

uniform-price auction 5
An auction in which all winning bidders pay the same price, typically the highest losing bid.

unilateral deviation 3
A change of strategy by a single player while all others keep their strategies fixed.

unrestricted domain 7
The requirement that a voting rule work for every possible profile of voter preferences.

unstable equilibrium 6
An equilibrium from which a small perturbation leads the dynamics away.

upper confidence bound (UCB) 13
A bandit algorithm that picks the action with the highest optimistic estimate of its value.

user equilibrium (Wardrop's 1st principle) 8
The flow in which each user takes a path that is no worse than any other, which is the Nash equilibrium of the routing game.

utilitarian criterion 12
A decision criterion that maximizes the sum of the utilities.

utilitarian welfare 11
The measure of social welfare given by the sum of all individuals' utilities.

utility 2
A numerical measure of how much a player values an outcome.

utility function 2
A function that assigns a utility to each outcome.

V

validator 14
A participant in a proof-of-stake blockchain who proposes and verifies blocks.

valuation 5
The maximum amount a bidder is willing to pay for an item or bundle.

valuation function (cake cutting) 11
A function that gives how much an agent values each piece of the cake.

value 5
The worth of an item or outcome to a participant.

value function (prospect theory) 12
A function in prospect theory that is concave for gains, convex for losses, and steeper for losses.

value of the game 2
The payoff a player can guarantee in a zero-sum game with optimal play.

veil of ignorance 12
A thought experiment in which principles of justice are chosen without knowing one's own position in society.

veto rule 7
A rule in which each voter vetoes one candidate and the candidate with the fewest vetoes wins.

Vickrey–Clarke–Groves (VCG) mechanism 4
A mechanism that chooses the welfare-maximizing outcome and charges each agent the externality they impose, which makes it truthful.

virtual value 5
The quantity of a bidder's value minus a correction for the information rent, used in optimal auctions.

virtual welfare 5
The sum of the virtual values of the winning bidders, which optimal auctions maximize.

von Neumann–Morgenstern axioms 1
The axioms of completeness, transitivity, continuity, and independence that imply preferences can be represented by expected utility.

vote splitting 7
The effect in which similar candidates divide the votes of their supporters and a less-preferred candidate wins.

W

Wardrop equilibrium 8
A flow in which all used paths between a pair of nodes have equal and minimal travel times.

water-filling 9
An allocation method that raises the lowest levels first, like water filling the lowest parts of a container.

weak budget balance 4
The requirement that a mechanism never run a deficit, though it may have a surplus.

weak dominance 3
The relation in which one strategy is never worse than another and is better against at least one opponent strategy.

weak equilibrium 2
An equilibrium in which some player has an alternative strategy that gives the same payoff.

weak notion of agency 10
The minimal definition of an agent as a system that is autonomous, social, reactive, and pro-active.

weakly dominant strategy 5
A strategy that is never worse than any other strategy and is strictly better against some strategy of the opponents.

weighted set packing 14
The problem of choosing disjoint sets that maximize total weight, the form of the winner determination problem.

weighted voting game 10
A game in which each player has a weight and a coalition wins if its total weight reaches a quota.

Wheatstone network 8
A five-node network with a bridge edge, used to show that adding capacity can worsen congestion.

Win-Stay Lose-Shift (WSLS, Pavlov) 13
A repeated-game strategy that repeats the previous action after a good outcome and switches after a bad one.

winner determination problem 14
The computational problem of choosing which bids to accept in a combinatorial auction to maximize revenue.

winner's curse 5
The tendency of the winner of a common-value auction to have overestimated the item's value and so to overpay.

winner-takes-all 6
A market in which a single firm or platform captures nearly all of the demand.

wisdom of crowds 12
The idea that the aggregated judgment of many individuals is often more accurate than that of any single expert.

worst-case equilibrium 9
The equilibrium with the highest cost or the lowest welfare among all equilibria.

Z

Zermelo's theorem 2
The theorem that in a finite two-player game of perfect information without chance, either one player has a winning strategy or both can force a draw.

zero-sum game 2
A game in which one player's gain is exactly the other's loss.

E

ε-approximate equilibrium 3
A strategy profile in which no player can improve their payoff by more than epsilon through a unilateral deviation.

ε-core 10
The set of allocations that no coalition can improve upon by more than epsilon.

ε-greedy 13
An action-selection rule that picks the best-known action with probability 1 minus epsilon and a random action otherwise.

APPENDIX B

Prerequisites for Algorithmic Game Theory

Algorithmic Game Theory · Prerequisites

This page collects the background that the 16 slide decks build on. The Chapter 0 · Preliminaries deck covers the most essential core: sets, counting, sums and the geometric series, argmax and derivatives, probability, Bayes' rule, U[0, 1], expected value, $x^T Ay$, best response, indifference and backward induction. Chapters 1–14 use the topics below. Each topic's card says what you need to know, in which chapters it is used, and includes two short self-check questions. The details open with the **Details and self-check** button. Click a question to see its answer.

The essential core. Sets and logic, expected value, conditional probability, reading a payoff matrix, argmax, Big-O notation and basic Python. The other topics are explained in the decks. Even so, the course moves faster if they are known in advance.

Terms. The TERMS line gives the terms used in the slides. The full list is in the course glossary.

Mathematics

Sets, proofs, counting, sums and logarithms, derivatives, convexity, matrices and fixed points.

Sets, functions and logic

Cartesian product · profiles · quantifiers

TERMS strategy set; strategy profile; unilateral deviation

Every game is a tuple of sets and functions: players N, strategy sets S_i, payoffs $u_i : S \to \mathbb{R}$. Equilibrium definitions are built from the patterns “for every …” and “there exists a … such that”.

CHAPTERS **Chapter 0** **Chapter 2** **Chapter 3** **Chapter 4** **Chapter 7**

- The set of strategy profiles is a Cartesian product: $S = S_1 \times S_2 \times \cdots \times S_n$. Its number of elements is $|S_1| \cdot |S_2| \cdot \ldots \cdot |S_n|$.

- s_{-i} denotes the strategies of everyone except player i. A profile is then written (s_i, s_{-i}).
- Nash equilibrium with quantifiers: for every player i and every $s_i' \in S_i$, $u_i(s_i^*, s_{-i}^*) \geq u_i(s_i', s_{-i}^*)$.
- In the negation the quantifiers swap: there exist a player i and a strategy s_i' with $u_i(s_i', s_{-i}^*) > u_i(s_i^*, s_{-i}^*)$. This is a profitable deviation.
- Relations: preferences are binary relations ($\succ$, $\succsim$). Voting theory relies on transitivity and completeness.

SELF-CHECK

▶ **Three players have 2, 3 and 4 pure strategies respectively. How many pure strategy profiles are there?**

▶ **What is the negation of the statement "for every player, no deviation is profitable"?**

Proof techniques

Direct proof · contradiction · induction · counterexample

TERMS backward induction; strategy-stealing argument

Most results in the course are claims about every deviation, every mechanism or every preference profile. That is why you need to be able to read and write short proofs.

CHAPTERS **Chapter 2** **Chapter 2 · Minimax and Alpha-Beta Pruning** **Chapter 3** **Chapter 4** **Chapter 7** **Chapter 9**

- Direct proof: show one by one that no unilateral deviation increases any player's payoff.
- Induction: backward induction and Zermelo's theorem proceed from the leaves of a finite game tree toward the root. Nim strategies are proved by induction on the position.
- Contradiction: the strategy-stealing argument shows that the first player cannot lose in Hex.
- Counterexample: a small game is enough to refute a general claim. For example, a vote profile that breaks monotonicity.
- Impossibility theorems (Arrow, Gibbard–Satterthwaite, Myerson–Satterthwaite) say that no rule satisfies a list of properties at the same time.

SELF-CHECK

▶ **How can the claim “every finite game has a Nash equilibrium in pure strategies” be refuted?**

▶ **Which proof technique does backward induction rest on?**

n!

Combinatorics and permutations

Product rule · n! · subsets · partitions

TERMS Shapley value; marginal contribution; Bell number

Strategy spaces, rankings, player orders and coalition structures grow very fast. Counting shows when brute-force search is hopeless.

CHAPTERS **Chapter 0** **Chapter 2** **Chapter 2 · Minimax and Alpha-Beta Pruning** **Chapter 3** **Chapter 7** **Chapter 10**

- n objects have n! orderings (permutations). A set of n elements has 2^n subsets. C(n, k) = n! / (k!(n − k)!).
- The Shapley value is the average of player i's marginal contributions over the n! arrival orders. Equivalent form: $\varphi_i = \Sigma\ (S \subseteq N\backslash\{i\})\ |S|!(n - |S| - 1)!/n! \cdot [v(S \cup \{i\}) - v(S)]$. This sum runs over 2^{n-1} coalitions.
- For m candidates there are m! strict rankings. This is why the Kemeny rule searches a very large space.
- The number of coalition structures (set partitions) of n players is the Bell number B_n: $B_3 = 5$, $B_4 = 15$, $B_{10} = 115{,}975$.

SELF-CHECK

▶ **To compute the Shapley value exactly in a game with 4 players, over how many arrival orders do you average?**

▶ **For n = 3, what weight does the formula give a coalition S with**

ln

Sums, logarithms and the exponential function

Geometric series · H_n · ln · e^x

TERMS discount factor; harmonic number; learning rate

Discounted repeated games, the Price of Stability, multiplicative weights and differential privacy are written with geometric sums, logarithms and the exponential function.

CHAPTERS **Chapter 0** **Chapter 2** **Chapter 2 · Minimax and Alpha-Beta Pruning** **Chapter 9** **Chapter 13** **Chapter 14**

- Geometric series: for $0 \le \delta < 1$, $\Sigma_{t \ge 0}\, \delta^t = 1/(1-\delta)$. The total value of a payoff c in every round is therefore $c/(1-\delta)$.
- Harmonic number $H_n = 1 + 1/2 + \ldots + 1/n \approx \ln n + 0.577$. In fair cost-sharing games the Price of Stability (PoS) is bounded by this number: $\text{PoS} \le H_n$.
- Logarithm rules: $\ln(ab) = \ln a + \ln b$, $\ln(a^k) = k \ln a$. Logarithmic growth is very slow ($\ln 10^6 \approx 13.8$).
- Exponential function: $e^x \ge 1 + x$ for every x; for small ε, $e^\varepsilon \approx 1 + \varepsilon$. Differential privacy guarantees are expressed with factors of e^ε.
- The multiplicative weights update has the form $w_i \leftarrow w_i \cdot e^{\eta \cdot g_i}$. Adding gains in the exponent means multiplying the weights.

SELF-CHECK

▶ **In a Prisoner's Dilemma with R = 3, T = 5, P = 1, for which values of δ does cooperation survive under the grim trigger strategy?**

▶ **What is H_4?**

f′

Derivatives and first-order conditions

Derivative · maximum · integral

TERMS maximize; minimize; first-order condition

Continuous strategies (quantities, bids, prices, flows) are optimized by setting the derivative to zero. Then you check that the point found is really a maximum.

CHAPTERS **Chapter 0** **Chapter 1** **Chapter 3** **Chapter 5** **Chapter 6** **Chapter 8**

- First-order condition: at an interior maximum of a differentiable f, $f'(x) = 0$. The second derivative ($f'' < 0$) or the endpoints are also checked, because $f' = 0$ can also indicate a minimum.
- The product rule $(fg)' = f'g + fg'$ and the power rule $(x^k)' = k\, x^{k-1}$ are enough for the derivations in the course.

- Cournot: each firm maximizes its profit $q_i \cdot P(q_1 + q_2) - \text{cost}$. Taking the derivative gives the best-response function.
- The integral as accumulated cost: the Beckmann potential $\Phi(f) = \Sigma_e \int_0^{f_e} \ell_e(t)\,dt$. The derivative of this potential with respect to f_e is the delay $\ell_e(f_e)$.

Example (Chapter 5). There are n bidders and the rivals bid $\beta \cdot v_j$, with $v_j \sim U[0, 1]$. Then a bid b wins with probability $(b/\beta)^{n-1}$. When $(v - b)(b/\beta)^{n-1}$ is maximized, the derivative vanishes when $(n - 1)(v - b) = b$. The result is $b = (n - 1)v/n$.

SELF-CHECK

▶ **What is the maximum of $f(q) = q(10 - q)$ for $q \geq 0$?**

▶ **What is $\int_0^x t\,dt$, and what is its derivative?**

Convex and concave functions

∪

Convex sets · Jensen · risk attitudes

TERMS convex; concave; Jensen's inequality

Convexity makes optimization easy: a local optimum is also a global optimum. Risk aversion is also explained by Jensen's inequality.

CHAPTERS Chapter 1 Chapter 3 Chapter 8 Chapter 12

- f is convex if $f(\lambda x + (1 - \lambda)y) \leq \lambda f(x) + (1 - \lambda)f(y)$ for every $\lambda \in [0, 1]$. For a twice differentiable f this condition means $f'' \geq 0$. f is concave if $-f$ is convex.
- A local minimum of a convex function is a global minimum. This is why the Wardrop equilibrium is well defined by the convex Beckmann potential.
- A set is convex if it contains the line segment between any two of its points. The set of mixed strategies (the simplex) is convex. Kakutani's theorem relies on this property.
- Jensen's inequality: if f is convex, $f(E[X]) \leq E[f(X)]$; if u is concave, $E[u(X)] \leq u(E[X])$. That is why a concave utility means risk aversion.

SELF-CHECK

▶ **Let $u(x) = \sqrt{x}$. A lottery pays 0 or 900 with probability 1/2 each. What are E[u] and the certainty equivalent?**

▶ **Is $f(x) = x^2$ convex, concave, or neither?**

A

Vectors and payoff matrices

Matrix indices · probability vectors · $x^T Ay$

TERMS payoff matrix; mixed strategy; support

Finite two-player games are a pair of matrices. Mixed strategies are probability vectors. Expected payoffs are found by matrix–vector multiplication.

CHAPTERS **Chapter 0** **Chapter 2** **Chapter 3** **Chapter 13**

- Two-matrix game (A, B): when row i meets column j, the row player gets A_{ij} and the column player gets B_{ij}. In a zero-sum game $B = -A$.
- A mixed strategy is a vector in the simplex $\Delta = \{ x : x_i \geq 0, \Sigma x_i = 1 \}$. Its support is the set of actions with $x_i > 0$.
- Expected payoffs: $(Ay)_i$ is the row player's payoff from pure action i against y. $x^T Ay$ is the payoff of x against y.
- Support enumeration solves small systems of linear equations: the payoffs on the support are equal and the probabilities sum to 1.

SELF-CHECK

▶ **Let A = [[3, 0], [5, 1]] (rows: cooperate, defect). For x = (1/2, 1/2) and y = (1, 0), what is $x^T Ay$?**

▶ **In the same matrix, does row D dominate row C?**

x*

Fixed points (Brouwer and Kakutani)

x = f(x) · existence · correspondences

TERMS fixed-point theorem; Nash's existence theorem; best-response correspondence

Nash's existence theorem is a fixed-point theorem. An equilibrium is a profile that is a best response to itself.

CHAPTERS **Chapter 3** **Chapter 6** **Chapter 13**

- A fixed point of f is a point satisfying $f(x) = x$. For a set-valued correspondence F, this condition becomes $x \in F(x)$.
- Brouwer: every continuous f mapping a nonempty, compact (closed and bounded) and convex set $K \subset \mathbb{R}^n$ into itself has a fixed point.

- Kakutani: the same result holds for a correspondence F whose values F(x) are nonempty and convex and whose graph is closed. Nash applies this theorem to the best-response correspondence on mixed profiles: $\sigma^* \in BR(\sigma^*)$.
- These theorems prove existence but give no efficient algorithm. The class PPAD measures exactly this gap.
- Dynamics such as the adoption model in Chapter 6 or the learning rules in Chapter 13 stop at fixed points of their update maps.

SELF-CHECK

▶ **What is the fixed point of f(x) = 1 − x on [0, 1]?**

▶ **Why does f(x) = x/2 have no fixed point on the open interval (0, 1)? Doesn't Brouwer's theorem rule this out?**

Probability and statistics

Distributions, expected value, Bayes' rule, order statistics, sampling and Markov models.

P

Probability and distributions

Events · independence · distribution function · uniform distribution

TERMS uniform distribution; common prior; independent private values

Mixed strategies, private types, bidder values and model errors are random variables. You need to read distributions and compute simple probabilities.

CHAPTERS **Chapter 0** **Chapter 1** **Chapter 3** **Chapter 4** **Chapter 5** **Chapter 6** **Chapter 12**

- Discrete distribution: probabilities $p_i \geq 0$ with $\Sigma p_i = 1$. Continuous distribution: a density $f \geq 0$ and a distribution function $F(x) = P(X \leq x)$.
- Uniform distribution U[0, 1]: on [0, 1], $F(x) = x$ and $f(x) = 1$. $E[X] = 1/2$, $Var(X) = 1/12$.
- Independence: $P(A \cap B) = P(A) \cdot P(B)$. Independent private values (IPV) means each bidder's value is drawn independently.
- The key step in auctions: for k independent U[0, 1] values, $P(\text{all} \leq b) = b^k$. This is the probability of beating k rivals who bid their values.

SELF-CHECK

▶ **Let X ~ U[0, 1]. What is P(X ≤ 0.3)?**

▶ **What is the probability that two independent U[0, 1] values are both at most 0.5?**

E

Expected value and variance

Linearity · E[f(X)] · spread

TERMS expected value; expected payoff; lottery

Mixed-strategy payoffs, expected revenue, bandit rewards and decision trees are all expected values.

CHAPTERS **Chapter 0** **Chapter 1** **Chapter 2** **Chapter 3** **Chapter 5** **Chapter 12** **Chapter 13**

- $E[X] = \Sigma\, x \cdot P(X = x)$ (discrete) or $\int x\, f(x)\, dx$ (continuous).
- Linearity: $E[aX + bY] = aE[X] + bE[Y]$. It holds even when X and Y are dependent.
- If f is not linear, in general $E[f(X)] \neq f(E[X])$ (Jensen's inequality in the convexity card).
- $Var(X) = E[X^2] - (E[X])^2$ measures the spread. Two lotteries with the same mean can have very different risk.
- The expected payoff is not the most likely payoff: a lottery that pays 100 with probability 0.3 never pays 30.

SELF-CHECK

▶ **A lottery pays 100 with probability 0.3 and 0 otherwise. What are E[X] and Var(X)?**

▶ **Does the equality E[X + Y] = E[X] + E[Y] require independence?**

P|

Conditional probability and Bayes' rule

Prior · likelihood · posterior · base rates

TERMS Bayes' rule; prior; posterior; base rate

Bayesian games, reputation systems, fairness metrics and decision analysis update beliefs with conditional probabilities.

CHAPTERS **Chapter 0** **Chapter 1** **Chapter 3** **Chapter 4** **Chapter 6** **Chapter 11**

- For $P(B) > 0$, $P(A \mid B) = P(A \cap B) / P(B)$.
- Bayes' rule: $P(H \mid E) = P(E \mid H) \cdot P(H) / P(E)$. The denominator comes from the law of total probability: $P(E) = P(E \mid H)P(H) + P(E \mid \neg H)P(\neg H)$.
- Fairness rates are conditional probabilities: the true positive rate $TPR = P(\hat{Y} = 1 \mid Y = 1)$, the false positive rate $FPR = P(\hat{Y} = 1 \mid Y = 0)$, the precision $PPV = P(Y = 1 \mid \hat{Y} = 1)$.
- Ignoring the base rate: if $P(H)$ is not taken into account, a positive result of an accurate test looks far more conclusive than it is.
- Sequential updating: today's posterior is tomorrow's prior.

SELF-CHECK

▶ **The prevalence is 1%, the sensitivity is 90% and the false positive rate is 5%. What is P(sick**

▶ **An independent second test is also positive. What is the updated probability?**

$v_{(1)}$

Order statistics for auctions

Highest · second highest · revenue

TERMS revenue equivalence theorem; winner's curse; virtual value

Auction revenue is the expected value of the highest or second-highest bid. That is why the expected value of the k-th largest of n random values appears throughout Chapter 5.

CHAPTERS **Chapter 5** **Chapter 14**

- For n independent $U[0, 1]$ values, the expectation of the k-th highest value is $(n + 1 - k)/(n + 1)$. The expectation of the highest value is $n/(n + 1)$, and that of the second highest is $(n - 1)/(n + 1)$.
- Revenue of the second-price auction = $E[\text{second-highest value}] = (n - 1)/(n + 1)$.
- Revenue of the first-price auction with the bid $b = (n - 1)v/n = (n - 1)/n \cdot n/(n + 1) = (n - 1)/(n + 1)$. As revenue equivalence predicts, the result is the same.
- The winner's curse: if estimation errors are distributed $U[-a, a]$, the expectation of the largest of n errors is $a(n - 1)/(n + 1)$. The winner is the bidder who overestimates the value the most.

- Virtual value $\varphi(v) = v - (1 - F(v))/f(v)$. For U[0, 1], $\varphi(v) = 2v - 1$ and the best reserve price is 1/2.

SELF-CHECK

▶ **With two bidders whose values are U[0, 1], what is the expected revenue of the second-price auction?**

▶ **With 4 bidders, what is the expected value of the winner?**

$\bar{x}$ Sampling, rates and simulation

Sample mean · error · confusion matrix

TERMS confusion matrix; true positive rate; sample complexity

Simulations, bandit estimates, fairness audits and learned mechanisms estimate quantities from noisy samples.

CHAPTERS **Chapter 5** **Chapter 11** **Chapter 13** **Chapter 14**

- Law of large numbers: the sample mean of N independent draws approaches the true mean. Its standard error is $\sigma/\sqrt{N}$.
- Confusion matrix rates: TPR = TP/(TP + FN), FPR = FP/(FP + TN), PPV = TP/(TP + FP), selection rate = (TP + FP)/total.
- In a simulation, the frequencies with which actions are seen estimate the mixed strategies. With small samples these frequencies fluctuate a lot.
- Sample complexity asks how many samples are needed to estimate to accuracy ε. With few samples, overfitting appears.

SELF-CHECK

▶ **By what factor must the sample grow to halve the standard error?**

▶ **TP = 40, FN = 10, FP = 10, TN = 40. What are TPR and FPR?**

γ Markov chains and Markov decision processes

States · transitions · rewards · discounting

TERMS policy; state–action value

Q-learning and repeated games evaluate long sequences of rewards. The model behind them is the Markov decision process.

CHAPTERS **Chapter 2** **Chapter 13**

- Markov property: the next state depends only on the present state (and action), not on the whole history.
- A Markov decision process (Markov decision process, MDP) consists of states S, actions A, transition probabilities $P(s' \mid s, a)$, rewards r and a discount $\gamma \in [0, 1)$. The goal is to maximize $E[\Sigma\, \gamma^t\, r_t]$.
- A policy maps states to actions. The greedy policy picks the action $\text{argmax}_a\, Q(s, a)$.
- Q-learning update: $Q(s, a) \leftarrow Q(s, a) + \alpha[r + \gamma \cdot \max_{a'} Q(s', a') - Q(s, a)]$. Here α is the learning rate.

SELF-CHECK

▶ **$Q(s, a) = 2$; $r = 1$; $\gamma = 0.9$; $\max Q(s', \cdot) = 4$; $\alpha = 0.5$. What is the new $Q(s, a)$?**

▶ **A reward of 1 is received at every step forever and $\gamma = 0.9$. What is the discounted total return?**

Optimization

argmax, linear programming and duality. This is the language of best responses, welfare and minimax.

arg

Optimization basics and argmax

Objective · constraints · max and argmax

TERMS optimization; social welfare; social optimum

Best responses, VCG allocations and social optima do the same job: among the feasible options, pick the one with the largest objective.

CHAPTERS **Chapter 0** **Chapter 3** **Chapter 4** **Chapter 5** **Chapter 9** **Chapter 10** **Chapter 12**

- An optimization problem has decision variables, an objective to maximize or minimize, and constraints that define the feasible set.
- max f is a value. argmax f is the set of maximizing points. In case of ties, argmax can contain more than one point. This is why mechanisms need a tie-breaking rule.
- Best response: $BR_i(s_{-i}) = \text{argmax}$ (over s_i) $u_i(s_i, s_{-i})$. VCG chooses the allocation $x^* = \text{argmax}_x\, \Sigma_i\, b_i(x)$.

- OPT denotes the best objective value. The Price of Anarchy compares the cost of an equilibrium with OPT.

SELF-CHECK

▶ **For $f(x) = -(x - 2)^2$ on [0, 5], what are max f and argmax f?**

▶ **The payoffs are a: 3, b: 5, c: 5. What is the argmax?**

LP

Linear programming and duality

Feasible polyhedron · dual · minimax

TERMS linear programming; minimax theorem; value of the game; saddle point

Zero-sum games, correlated equilibria and some mechanism design problems are solved by linear programs. The reason the maximin value equals the minimax value is LP duality.

CHAPTERS **Chapter 3** **Chapter 13** **Chapter 14**

- An LP maximizes or minimizes a linear objective under linear inequality constraints. The feasible set is a convex polyhedron. If the feasible set is a nonempty, bounded polyhedron, at least one optimum is at a vertex.
- LPs are solved in polynomial time (ellipsoid and interior-point methods). The simplex method is fast in practice but exponential in the worst case.
- Duality: every maximization LP has a minimization dual. Weak duality: primal value ≤ dual value. Strong duality: at the optimum the two are equal.
- The row player's LP in a zero-sum game: maximize v subject to $\Sigma_i\, x_i A_{ij} \geq v$ for every column j, $\Sigma x_i = 1$ and $x \geq 0$. The column player's LP is its dual. Hence max min = min max (von Neumann's minimax theorem).
- Correlated and coarse correlated equilibria are also LP feasibility problems. That is why they can be computed in polynomial time.

SELF-CHECK

▶ **Let A = [[3, 1], [4, 2]] (the row player maximizes). What are the maximin, the minimax and the saddle point?**

▶ **In Matching Pennies (payoffs ±1), what are the value of the game and the best strategy?**

Algorithms and complexity

Running time, algorithm design, XOR, graphs and the classes P, NP, PPAD and PLS.

O Big-O and running time

Growth rates · polynomial and exponential

TERMS greedy; alpha–beta pruning

Algorithmic game theory is not content with whether a result exists. It also asks whether that result can be computed quickly.

CHAPTERS **Chapter 2** **Chapter 2 · Minimax and Alpha-Beta Pruning** **Chapter 4** **Chapter 8** **Chapter 9** **Chapter 10**

- Common growth rates: $O(1) < O(\log n) < O(n) < O(n \log n) < O(n^2) < O(n^3) < O(2^n) < O(n!)$. O is an upper bound, and Θ is a tight bound.
- Polynomial time ($O(n^k)$) is the working definition of the word "efficient".
- A game tree with branching factor b and depth d has about b^d leaves. If moves are examined in the best order, alpha-beta pruning looks at about $b^{d/2}$ leaves.
- Examples in the course: Gale–Shapley $O(n^2)$, the Hungarian algorithm $O(n^3)$, Dijkstra with a binary heap $O((V + E) \log V)$.

SELF-CHECK

▶ **Assigning 10 robots to 10 tasks: how do brute force and the Hungarian algorithm compare?**

▶ **Is $O(n \log n)$ polynomial?**

⟳ Algorithm design and step-by-step tracing

Recursion · greedy · dynamic programming

TERMS deferred acceptance

Many of the slide decks contain step-by-step code traces. You need to be able to trace an algorithm, state its input and output, and justify that it halts and is correct.

CHAPTERS **Chapter 0** **Chapter 2** **Chapter 2 · Minimax and Alpha-Beta Pruning** **Chapter 4** **Chapter 10**

- Recursion: minimax computes a node from the values of its children. Backward induction is the same idea on a game tree.

- Dynamic programming stores solved subproblems. For example, the winning and losing positions of a subtraction game.
- Greedy algorithms make the locally best choice at each step. They are fast but do not always give the optimum. The Hungarian algorithm, however, finds the optimal assignment.
- Gale–Shapley (deferred acceptance): proposers propose in order of preference, and receivers hold on to the best offer. The algorithm ends with a stable matching that is best for the proposing side.

SELF-CHECK

▶ **Why does Gale–Shapley terminate?**

▶ **The minimax leaves under a MAX node are 3, 7 and 2. What is the value of the node?**

b^d

Trees, recursion and exponential growth

Root · leaf · depth · branching factor · $\log_b$

TERMS game tree; branching factor; depth-first search; iterative deepening

Minimax and alpha-beta traverse the game tree recursively, in depth-first order. The size of the tree grows exponentially with the branching factor and depth. How deep one can search in a given time is computed with a logarithm.

CHAPTERS **Chapter 2** **Chapter 2 · Minimax and Alpha-Beta Pruning**

- Tree terms: the root is the starting position, the leaves are game endings or positions at the depth limit. The depth of a node is its distance from the root. In a game tree the layers belong alternately to the MAX and MIN players.
- If every internal node has b children, there are b^d leaves at depth d and $1 + b + \ldots + b^d = (b^{d+1} - 1)/(b - 1)$ nodes in total. The vast majority of nodes are in the last layer: about 90% of them for b = 10.
- In a recursive traversal, the value of a node is computed from the values of its children. Only the path from the root to the current node is held in memory. So the time is $O(b^d)$ while the memory is only $O(b \cdot d)$.
- Logarithm: with a budget of N leaves, the reachable depth is $d = \log_b N = \ln N / \ln b$. Since $b^{d/2} = (\sqrt{b})^d$, with the best move ordering alpha-beta reaches about twice the depth with the same budget.

- Iterative deepening repeats the search for d = 1, 2, 3, ... The total cost of the earlier rounds is only about $1/(b-1)$ times that of the last round.

SELF-CHECK

▶ **In a uniform tree with b = 3 and d = 2, how many nodes and how many leaves are there?**

▶ **In chess b ≈ 35, and a program evaluates 10^6 leaves per second. How many layers deep can it search in 10 seconds?**

⊕

Binary numbers and XOR

Bit operations · Nim

TERMS nim-sum; impartial game; Sprague–Grundy theorem

Nim is solved exactly by the bitwise XOR of the pile sizes. The Sprague–Grundy theory extends this result to other impartial games.

CHAPTERS **Chapter 2** **Chapter 2 · Minimax and Alpha-Beta Pruning**

- XOR (⊕) adds binary digits without carry: a bit of the result is 1 if the number of inputs with a 1 in that position is odd.
- $x \oplus x = 0$ and $x \oplus 0 = x$. XOR is commutative and associative.
- Nim (normal play): if the nim-sum of the piles is 0, the player to move loses when both sides play best.

SELF-CHECK

▶ **What is $3 \oplus 4 \oplus 5$?**

▶ **The piles are 1, 2 and 3. Who wins under best play?**

NP

P, NP, PPAD and PLS

Reductions · hardness · total search problems

TERMS NP-hard; PPAD-complete; PLS-complete; ε-approximate equilibrium

Complexity classes explain which objects in game theory are easy to compute (equilibria of zero-sum games, correlated equilibria) and which are believed to be hard (general Nash equilibria, optimal allocations).

CHAPTERS **Chapter 3** **Chapter 4** **Chapter 7** **Chapter 8** **Chapter 14**

- P: problems solvable in polynomial time. NP: problems for which a proposed solution can be verified in polynomial time.
- NP-hard: every NP problem reduces to it in polynomial time. NP-complete = both NP-hard and in NP. Examples in the course: the argmax step of VCG in combinatorial auctions, the Kemeny ranking, manipulating some voting rules.
- Why isn't finding a Nash equilibrium NP-complete? An equilibrium always exists, so the yes/no question is trivial. The hardness lies in finding the equilibrium.
- PPAD: search problems whose solution is guaranteed by a parity (end-of-the-line) argument. Brouwer, Sperner and Nash are in this class. Two-player Nash equilibrium is PPAD-complete (Chen–Deng, 2006; journal version Chen–Deng–Teng, 2009).
- PLS: local search problems. Finding a pure equilibrium of a congestion game is PLS-complete.

SELF-CHECK

▶ **Can a correlated equilibrium be computed in polynomial time?**

▶ **Why is "Does this game have a mixed Nash equilibrium?" not an interesting NP question?**

G

Graphs, shortest paths and flows

Directed graphs · cycles · flows · load

TERMS graph; node; edge; latency function

Routing games, network formation, matching markets, voting tournaments and kidney exchange are all graph problems.

CHAPTERS **Chapter 4** **Chapter 7** **Chapter 8** **Chapter 9** **Chapter 10**

- Graphs consist of nodes and (directed or undirected) edges. They are stored as an adjacency list or an adjacency matrix. Basic concepts: path, cycle, connectivity, bipartite graph.
- Shortest paths: Dijkstra requires nonnegative weights. Bellman–Ford allows negative weights (provided there is no negative cycle).
- Flows: a flow of r units from s to t is split over paths. The load of an edge is the sum of the flows of the paths that use it. Flow is conserved at every internal node.

- In routing games the edge costs are not fixed weights but load-dependent latency functions $\ell_e(x)$. This is what gives rise to congestion and Braess's paradox.
- Cycles matter elsewhere too: Top Trading Cycles (TTC) extracts the cycles of a pointer graph. A majority graph containing a cycle is the Condorcet paradox.

SELF-CHECK

▶ **Pigou network: one unit of flow, edge costs 1 and $\ell(x) = x$. What are the equilibrium cost, the optimal cost and the PoA?**

▶ **Does Dijkstra work with negative edge weights?**

Strategic reasoning and economics

Expected utility, best responses, welfare terms and regret.

E[u]

Expected utility and mixed strategies

Lotteries · indifference · risk attitude

TERMS expected utility; indifference principle; risk-neutral; certainty equivalent

When there is chance, a player compares lotteries by their expected utility $\Sigma\, p(x)\cdot u(x)$. Mixed equilibria are computed this way, and risk attitudes are modeled this way too.

CHAPTERS **Chapter 0** **Chapter 1** **Chapter 3** **Chapter 5** **Chapter 12**

- The expected utility of a lottery is $\Sigma\, p(x)\cdot u(x)$. For a risk-neutral player, $u(x) = x$. Risk aversion means a concave u.
- The indifference principle: in a mixed Nash equilibrium, every action in a player's support yields the same expected payoff. Actions outside the support do not yield more.
- So each player's mixing probabilities are chosen to make the *other* player indifferent. A player's own payoffs determine the opponent's mixture.
- Most auction results assume that bidders are risk neutral (Chapter 5). Risk attitudes are the subject of Chapter 1.

SELF-CHECK

▶ **The column player plays Left with probability q. The row player's payoffs are: Top = 2q, Bottom = 1 − q. For which q is the row player willing to mix?**

▶ **In a mixed equilibrium, can an action outside the support yield strictly more than the actions in the support?**

BR

Best responses and dominance

Best response · dominance · deviation

TERMS best response; dominant strategy; dominated strategy; Nash equilibrium

The core skill of the course is this: given the others' choices, what is each player's best response, and does anyone want to deviate?

CHAPTERS **Chapter 0** **Chapter 2** **Chapter 3** **Chapter 8** **Chapter 12**

- Best response: the action or actions that maximize a player's payoff while the others' actions are fixed.
- Strict dominance: s_i is strictly better than s_i' in every case. A strictly dominated strategy is not played in any Nash equilibrium, not even with positive probability.
- Weak dominance: never worse, sometimes strictly better. Weakly dominated strategies can still appear in equilibria. The result of eliminating them can depend on the order of elimination.
- If every player best-responds to the others, the profile is a Nash equilibrium. Individual stability does not guarantee the social optimum (Prisoner's Dilemma).

SELF-CHECK

▶ **In the Prisoner's Dilemma, is (Cooperate, Cooperate) a Nash equilibrium?**

▶ **Can a strictly dominated strategy be in the support of a mixed equilibrium?**

€

Economics terms

Valuation · utility · welfare · revenue

TERMS valuation; quasi-linear utility; revenue; surplus

Mechanism design and auctions use a small set of economics terms again and again. Confusing these terms, especially welfare and revenue, is a common mistake.

CHAPTERS **Chapter 4** **Chapter 5** **Chapter 6** **Chapter 9**

- Valuation v_i: the value of an outcome to agent i. Quasi-linear utility: $u_i = v_i(x) - p_i$, that is, value minus payment.
- Social welfare: $\Sigma_i\ v_i(x)$. Payments are transfers between the agents and the seller. They therefore cancel out in welfare.
- Revenue: $\Sigma_i\ p_i$, the amount the seller collects. Welfare = the agents' total utility + revenue.
- Surplus (gains from trade) in a market is the difference between the buyer's value and the seller's cost in the trades that take place.
- An externality is the effect of one agent's action on the others' payoffs. The agent does not pay for this effect (congestion, network effects).

SELF-CHECK

▶ **A bidder with value 10 wins and pays 7. What are the utility, the revenue and the welfare?**

▶ **Can a mechanism raise revenue while lowering welfare?**

R

Regret and online learning

External regret · Hedge · bandits

TERMS regret; no-regret; multiplicative weights; multi-armed bandit

Learning in games measures an algorithm by its regret: how much worse it did, in hindsight, than the best single action.

CHAPTERS **Chapter 1** **Chapter 9** **Chapter 13**

- External regret after T rounds: $R_T = \max_a \Sigma_t\ g_t(a) - \Sigma_t\ g_t(a_t)$. If $R_T / T \to 0$, the algorithm is no-regret.
- With n actions and payoffs in [0, 1], multiplicative weights (Hedge) gives $R_T \le \ln n/\eta + \eta T$. Choosing $\eta = \sqrt{(\ln n / T)}$ gives $R_T \le 2\sqrt{(T \ln n)}$.
- The follow-the-leader algorithm can be forced to switch constantly and suffers linear regret. Randomization fixes this problem.
- If all players use no-regret learning, the time average of the game converges to the coarse correlated equilibria. The PoA bounds for smooth games remain valid.
- In bandit problems only the reward of the chosen action is seen. UCB adds an exploration bonus of size $\sqrt{(2 \ln t / N_a)}$.

- The minimax regret (Savage) in Chapter 1 carries the same word but is a different criterion, used for a single decision.

SELF-CHECK

▶ **Hedge with n = 2 actions and T = 100 rounds: what is the bound on total and average regret?**

▶ **Why does an average regret going to 0 not mean that the learner played the best action in every round?**

Programming

Every deck contains short Python programs and code traces.

py

Programming with Python

Loops · functions · lists and dictionaries · random

Every deck contains short Python programs and code traces: payoff tables, auctions, Shapley values, bandits, Q-learning. You need to be able to read, run and modify these programs.

CHAPTERS **Chapter 1** **Chapter 2** **Chapter 3** **Chapter 4** **Chapter 5** **Chapter 6** **Chapter 7** **Chapter 8** **Chapter 9** **Chapter 10** **Chapter 11** **Chapter 12** **Chapter 13** **Chapter 14**

- The core of the language: variables, loops, functions, recursion, lists, tuples, dictionaries and list comprehensions.
- Payoff matrices as nested lists (A[i][j]), strategy profiles with itertools.product, player orders with itertools.permutations.
- Randomness: random.random(), random.choice, and seeded generators for reproducible simulations (random.Random(3)).
- Math functions: math.exp, math.log, math.sqrt (multiplicative weights, UCB).
- Experiments: a simulation is repeated many times, the results are averaged and compared with the theoretical prediction.

SELF-CHECK

▶ **How many tuples does itertools.product(range(2), repeat=3) produce?**

▶ **How many orders does itertools.permutations(range(3)) produce?**

No topic matches the search or the filters.

▶ Prerequisites chapter by chapter

© Sercan Külcü · Algorithmic Game Theory

APPENDIX C

Important People in Algorithmic Game Theory

Algorithmic Game Theory · People

Short profiles, gathered on 62 cards, of the mathematicians, economists and computer scientists behind the results used in the course. The list runs from Borda and Condorcet to the Price of Anarchy and PPAD. Each card links to the course chapters where the person's work appears; the details open with the button at the bottom of the card.

JB

Jean-Charles de Borda

1733–1799 · French · naval officer and mathematician, Académie des sciences

Borda count · 1770

Proposed the Borda count. In this positional voting rule, the options in each ranking receive m−1, m−2, …, 0 points.

positional scoring | ranked ballot | Condorcet criterion

CHAPTERS **Chapter 7 · Social Choice and Voting**

Significance: The Borda count is the classic scoring rule. It never elects the Condorcet loser, but it can fail to elect the Condorcet winner.

Borda presented his method to the Academy in 1770, and it was published in 1784. Strong rivals can be buried at the bottom of the ranking, so the rule is easy to manipulate. This weakness was later made precise by the Gibbard–Satterthwaite theorem.

MC

Marquis de Condorcet

1743–1794 · French · philosopher and mathematician

Condorcet paradox · jury theorem · 1785

In his 1785 *Essai* he showed that pairwise majority voting can cycle: A beats B, B beats C, and C beats A.

pairwise majority | Condorcet winner | jury theorem

CHAPTERS **Chapter 7 · Social Choice and Voting**

Significance: The Condorcet winner and the voting paradox are the starting point of social choice theory and of Arrow's impossibility theorem.

According to the jury theorem, if each voter independently decides correctly with probability $p > 1/2$, the probability that the majority decision is correct approaches 1 as the number of voters grows. Condorcet died in prison during the French Revolution.

CB

Charles L. Bouton

1869–1922 · American · Harvard University

Complete solution of Nim · 1901–1902

In the paper "Nim, a game with a complete mathematical theory" (*Annals of Mathematics*, 1901–1902) he solved Nim completely. The player to move loses under best play if the binary XOR sum of the pile sizes (the nim-sum) is 0.

Nim · nim-sum · P- and N-positions

CHAPTERS **Chapter 2 · Games and Representations** **Chapter 2 · Minimax and Alpha-Beta Pruning**

Significance: Bouton's rule gives, with a single XOR operation, the result that minimax finds by traversing the whole game tree. It is among the first complete results of combinatorial game theory.

The winning move is the one that makes the nim-sum 0. Sprague (1935) and Grundy (1939) extended this idea to all impartial games.

ÉB

Émile Borel

1871–1956 · French · Université de Paris (Minister of the Navy in 1925)

Mixed strategies · 1921

In notes published between 1921 and 1927 he treated strategic games with mixed (randomized) strategies (mixed strategies).

mixed strategy · zero-sum game · randomization

CHAPTERS **Chapter 2 · Games and Representations**

Significance: Borel posed the minimax question for symmetric zero-sum games. Von Neumann proved the general minimax theorem in 1928.

Borel solved small cases but doubted that the minimax property holds in every game. The priority dispute between Borel and von Neumann was revisited in Econometrica in 1953.

EZ

Ernst Zermelo

1871–1953 · German · Göttingen, Zurich, Freiburg

Zermelo's theorem on chess · 1913

The first formal theorem of game theory is attributed to him (1913). According to the theorem, in chess either White can force a win, or Black can force a win, or both sides can guarantee at least a draw.

perfect information · determinacy · backward induction

CHAPTERS **Chapter 2 · Games and Representations** · **Chapter 2 · Minimax and Alpha-Beta Pruning**

Significance: The result is the foundation of backward induction (backward induction) and of the determinacy of finite two-player games of perfect information.

Zermelo is better known for his axioms of set theory (1908). His chess paper is the ancestor of the game tree search used for Tic-Tac-Toe, Nim and Connect Four in the course labs.

AP

Arthur Cecil Pigou

1877–1959 · British · University of Cambridge

Pigou's example · externalities · 1912/1920

He described two roads, one that can become congested and one with a fixed delay. Selfish drivers overuse the congestible road (*Wealth and Welfare*, 1912; *The Economics of Welfare*, 1920).

externality · Pigouvian tax · congestion

CHAPTERS **Chapter 8 · Network Design and Routing** · **Chapter 9 · The Price of Anarchy**

Significance: Pigou's example is the standard example of selfish routing. With linear delays the price of anarchy (price of anarchy) is 4/3. Roughgarden and Tardos proved that this is the worst case.

Pigou proposed taxing activities that create negative externalities. Congestion pricing (a marginal-cost toll) is the network counterpart of this idea and makes the Wardrop equilibrium optimal.

HS

Hugo Steinhaus

1887–1972 · Polish · universities of Lwów and Wrocław

Fair cake cutting · 1948

He began the mathematical theory of fair division with the paper “The Problem of Fair Division” (Econometrica, 1948).

proportionality · cut-and-choose · cake cutting

CHAPTERS **Chapter 11 · Algorithmic Fairness and Ethics**

Significance: He gave a proportional method for three players. For n players he also announced the Banach–Knaster last diminisher (last diminisher) protocol.

Fair division looks for two kinds of allocation: proportional (everyone gets at least 1/n by their own valuation) or envy-free. The algorithmic theory was later developed by Brams, Taylor, Procaccia and others.

OM

Oskar Morgenstern

1902–1977 · German-born Austrian-American · Princeton University

Theory of Games and Economic Behavior · 1944

With von Neumann he wrote *Theory of Games and Economic Behavior* (1944). This book established game theory as a separate field within economics.

expected utility · strategic form · rational choice

CHAPTERS **Chapter 1 · Decision Theory** **Chapter 2 · Games and Representations**

Significance: The von Neumann–Morgenstern expected utility axioms justify modeling players as decision makers who maximize expected payoff.

The utility axioms (completeness, transitivity, continuity, independence) were added in the second edition of 1947. The behavioral work of Allais, Kahneman and Tversky later put these axioms to the test.

JvN

John von Neumann

1903–1957 · Hungarian-American · Institute for Advanced Study, Princeton

Minimax theorem · 1928

He proved the minimax theorem for two-player zero-sum games (1928). With Morgenstern he wrote *Theory of Games and Economic Behavior* (1944).

zero-sum game · mixed strategy · saddle point

CHAPTERS **Chapter 1 · Decision Theory** **Chapter 2 · Games and Representations** **Chapter 2 · Minimax and Alpha-Beta Pruning** **Chapter 3 · Nash Equilibrium**

Significance: Minimax links strategic optimization to duality in linear programming and to adversarial search.

In every finite two-player zero-sum game, the max-min value with mixed strategies equals the min-max value. The 1944 book introduced cooperative games (stable sets), and the 1947 edition added the expected utility axioms.

WV

William Vickrey

1914–1996 · Canadian-American · Columbia University

Second-price auction · 1961

He studied auctions in the paper "Counterspeculation, Auctions, and Competitive Sealed Tenders" (1961). He showed that in a sealed-bid second-price auction, bidding one's own value is a dominant strategy (dominant strategy).

Awards: Nobel Prize in Economics 1996 (with James Mirrlees); he died three days after the prize was announced

second-price auction | dominant strategy | revenue equivalence

CHAPTERS **Chapter 4 · Mechanism Design** **Chapter 5 · Auctions** **Chapter 6 · Online Platforms**

Significance: The Vickrey auction is the first VCG mechanism and the template for truthful mechanism design.

The same paper foreshadowed the revenue equivalence of standard auction formats. Vickrey also advocated congestion pricing for roads and public transport.

HS

Herbert A. Simon

1916–2001 · American · Carnegie Mellon University

Bounded rationality · satisficing · 1955

He introduced the concept of bounded rationality (bounded rationality). Real decision makers have limited information and computational power, so they look not for the best option but for one that is good enough (satisficing).

Awards: Turing Award 1975 (with Allen Newell); Nobel Prize in Economics 1978

bounded rationality | satisficing | heuristics

CHAPTERS **Chapter 3 · Nash Equilibrium** **Chapter 12 · Behavioral and Collective Decisions**

Significance: Bounded rationality highlights the computational limits of agents. The topic comes up often in algorithmic game theory and in multi-agent artificial intelligence.

The paper "A Behavioral Model of Rational Choice" (1955) argued that full optimization is unrealistic because of search costs. Simon is among the founders of artificial intelligence (Logic Theorist, 1956).

CS

Claude E. Shannon

1916–2001 · American · Bell Labs, MIT

Programming chess · 1950

In the paper "Programming a Computer for Playing Chess" (*Philosophical Magazine*, 1950) he outlined a chess-playing program: depth-limited minimax search with an evaluation function at the leaves.

Awards: Kyoto Prize 1985

evaluation function | depth limit | Shannon number

CHAPTERS **Chapter 2 · Minimax and Alpha-Beta Pruning**

Significance: His rough estimate of the number of possible chess games (about 10^{120}) is the standard figure showing why exhaustive brute-force search is impossible, and it is known as the Shannon number.

In the paper he distinguished full-width search (type A), which examines every move to the same depth, from selective search (type B), which examines only promising moves. He is also the author of "A Mathematical Theory of Communication" (1948), which founded information theory.

GB

George W. Brown

1917–2005 · American · RAND, UCLA, UC Irvine

Fictitious play · 1951

He proposed fictitious play (fictitious play) (1951). In this dynamic, each player best-responds to the observed frequencies of the opponent's past moves.

fictitious play | best response | observed frequencies

CHAPTERS **Chapter 13 · Learning and Adaptation**

Significance: Fictitious play is the first learning dynamic for games. In 1951 Julia Robinson proved that it converges in two-player zero-sum games.

Shapley (1964) gave a 3×3 game in which fictitious play cycles. This showed that simple learning need not reach a Nash equilibrium in general games.

LH

Leonid Hurwicz

1917–2008 · Polish-American (born in Moscow) · University of Minnesota

Mechanism design · incentive compatibility · 1972

He founded the theory of mechanism design and introduced the concept of incentive compatibility (incentive compatibility) (1972).

Awards: Nobel Prize in Economics 2007 (with Eric Maskin and Roger Myerson); at 90, the oldest laureate up to that time

incentive compatibility | decentralization of information | implementation

CHAPTERS **Chapter 4 · Mechanism Design**

Significance: Mechanism design turns game theory around: the rules are chosen so that self-interested behavior produces the desired outcome.

Hurwicz showed that in exchange economies no mechanism that is efficient and individually rational can be incentive compatible in dominant strategies. This result framed the trade-offs the field studies.

BH

David Blackwell & James Hannan

1919–2010 · 1922–2010 · American · UC Berkeley (Blackwell); Michigan State University (Hannan)

Approachability 1956 · Hannan consistency 1957

They laid the foundations of no-regret (no-regret) learning. Hannan (1957) constructed a strategy whose external regret in repeated games goes to zero over time. Blackwell (1956) proved the approachability (approachability) theorem for vector payoffs.

Awards: Blackwell: first Black member elected to the US National Academy of Sciences (1965); von Neumann Theory Prize 1979

external regret | approachability | repeated game

CHAPTERS **Chapter 13 · Learning and Adaptation**

Significance: No-regret algorithms such as Hedge and regret matching derive from these results and converge to coarse correlated equilibria.

Blackwell's theorem determines which convex sets a player can force the average vector payoff to approach. Hannan consistency and calibration follow as consequences of this theorem.

JH

John Harsanyi

1920–2000 · Hungarian-American · UC Berkeley

Bayesian games · 1967–68

He developed the theoretical framework for games with incomplete information ("Games with Incomplete Information Played by Bayesian Players", 1967–68).

Awards: Nobel Prize in Economics 1994 (with John Nash and Reinhard Selten)

type spaces · common prior · Bayes–Nash equilibrium

CHAPTERS **Chapter 3 · Nash Equilibrium** **Chapter 4 · Mechanism Design**

Significance: Bayesian games (Bayesian games) model private types (values, costs) drawn from a common prior (common prior). Auction and mechanism theory rest on this model.

The Harsanyi transformation adds a chance move ("Nature") that draws each player's type. Incomplete information thus becomes imperfect information. The purification theorem (1973) gives a new interpretation of mixed strategies.

JM

John Maynard Smith

1920–2004 · British · University of Sussex

Evolutionarily stable strategy · 1973

With George Price he introduced the evolutionarily stable strategy (evolutionarily stable strategy, ESS) ("The Logic of Animal Conflict", Nature, 1973).

Awards: Crafoord Prize 1999 (with Ernst Mayr and George C. Williams)

ESS · Hawk–Dove · replicator dynamics

CHAPTERS **Chapter 13 · Learning and Adaptation**

Significance: The ESS and the replicator dynamics (Taylor and Jonker, 1978) carry game theory to populations that do not assume rational players.

A strategy adopted by a population is evolutionarily stable if a small group of mutants cannot invade it. Every ESS is a symmetric Nash equilibrium. Book: *Evolution and the Theory of Games* (1982).

DG

David Gale

1921–2008 · American · UC Berkeley

Deferred acceptance · 1962

With Shapley he discovered the deferred acceptance (deferred acceptance) algorithm ("College Admissions and the Stability of Marriage", 1962).

stable matching | blocking pair | deferred acceptance

CHAPTERS **Chapter 4 · Mechanism Design** **Chapter 5 · Auctions**

Significance: Deferred acceptance always finds a stable matching with $O(n^2)$ proposals. The matching it finds is the best for the proposing side, and the algorithm is strategy-proof (strategy-proof) for that side.

Gale died in 2008; the theory of stable allocations was honored with the Nobel Prize in Economics in 2012. He also contributed to duality in linear programming and to the game of Hex. Gale gave one of the known proofs that a game of Hex cannot end in a draw and tied this result to the Brouwer fixed-point theorem.

KA

Kenneth Arrow

1921–2017 · American · Stanford University

Arrow's impossibility theorem · 1951

In the book *Social Choice and Individual Values* (1951) he proved that no rule for aggregating rankings satisfies a few natural axioms at once.

Awards: Nobel Prize in Economics 1972 (with John Hicks)

preference aggregation | IIA | no dictator

CHAPTERS **Chapter 2 · Games and Representations** **Chapter 4 · Mechanism Design** **Chapter 7 · Social Choice and Voting**

Significance: With three or more options, every rule that accepts every preference ranking (universal domain), outputs a complete, transitive ranking as the social outcome, and satisfies unanimity (Pareto) and independence of irrelevant alternatives (independence of irrelevant alternatives, IIA) is a dictatorship.

Arrow's theorem began axiomatic social choice. Arrow also proved, with Debreu, the existence of a competitive equilibrium (Arrow–Debreu, 1954). This proof relies on fixed-point arguments similar to Nash's.

TS

Thomas Schelling

1921–2016 · American · Harvard University, University of Maryland

Focal points · commitment · segregation model

In the book *The Strategy of Conflict* (1960) he studied conflict and coordination. He introduced focal (Schelling) points (focal points) and the concept of credible commitment.

Awards: Nobel Prize in Economics 2005 (with Robert Aumann)

focal point · commitment · individual motives and aggregate behavior

CHAPTERS **Chapter 2 · Games and Representations** · **Chapter 3 · Nash Equilibrium** · **Chapter 10 · Multi-Agent Systems**

Significance: The segregation model (1969–71) showed that mild individual preferences can produce strong aggregate segregation. This model is among the first examples of agent-based simulation.

Schelling showed that limiting one's own options (burning bridges) can have strategic value. This idea appears often in deterrence and bargaining theory.

JW

John Wardrop

1922–1989 · British · Road Research Laboratory, University College London

Wardrop equilibrium · 1952

He proposed two principles for traffic assignment (1952): user equilibrium and system optimum. These principles are now known by his name.

user equilibrium · system optimum · traffic assignment

CHAPTERS **Chapter 8 · Network Design and Routing** · **Chapter 9 · The Price of Anarchy**

Significance: In a Wardrop equilibrium, every used route has the same, minimal delay. The price of anarchy compares this equilibrium with the system optimum.

Beckmann, McGuire and Winsten (1956) showed that Wardrop equilibria minimize a potential function. Equilibria can therefore be computed by convex optimization.

LS

Lloyd Shapley

1923–2016 · American · RAND Corporation, UCLA

Shapley value · stochastic games · matching

He introduced the Shapley value (1953) and stochastic games (1953). With Gale he discovered the deferred acceptance algorithm (1962).

Awards: Nobel Prize in Economics 2012 (with Alvin Roth)

Shapley value | coalition | core

CHAPTERS **Chapter 4 · Mechanism Design** **Chapter 5 · Auctions** **Chapter 10 · Multi-Agent Systems**

Significance: The Shapley value is the only allocation rule that satisfies the axioms of efficiency, symmetry, additivity and the dummy player together.

This value is the average of a player's marginal contribution over all arrival orders. Today it is used in cost sharing and in explaining machine learning predictions (SHAP). The Bondareva–Shapley theorem characterizes the games with a nonempty core.

HK

Harold W. Kuhn

1925–2014 · American · Princeton University

Kuhn's theorem · 1953 · Hungarian method · 1955

In the paper "Extensive games and the problem of information" (1953) he defined extensive-form games with information sets as we know them today. He showed that pure strategies suffice in finite games of perfect information and that backward induction yields an equilibrium.

Awards: John von Neumann Theory Prize 1980 (with David Gale and Albert W. Tucker)

extensive form | information set | Hungarian method

CHAPTERS **Chapter 2 · Games and Representations** **Chapter 2 · Minimax and Alpha-Beta Pruning** **Chapter 10 · Multi-Agent Systems**

Significance: In Kuhn's framework, Zermelo's chess result becomes a general theorem. The value in the minimax tree is the pure-strategy value of the game of perfect information.

He published the Hungarian method for the assignment problem in 1955. The Karush–Kuhn–Tucker conditions also bear his name. He was a close friend and colleague of Nash.

JN

John Nash

1928–2015 · American · Princeton University, MIT

Nash equilibrium · 1950–51

He defined the Nash equilibrium and proved that every finite game has an equilibrium in mixed strategies (PNAS 1950; Annals of Mathematics 1951).

Awards: Nobel Prize in Economics 1994 (with John Harsanyi and Reinhard Selten); Abel Prize 2015 (with Louis Nirenberg)

best response · fixed point · bargaining solution

CHAPTERS **Chapter 2 · Games and Representations** **Chapter 3 · Nash Equilibrium**

Significance: The Nash equilibrium is the central solution concept of noncooperative game theory. The existence proof rests on a fixed-point theorem.

The 1950 proof uses Kakutani's fixed-point theorem and the 1951 proof uses Brouwer's. The proof is not constructive; it was shown much later that computing an equilibrium is PPAD-complete. Nash also introduced the Nash bargaining solution (1950).

RS

Reinhard Selten

1930–2016 · German · Universität Bonn

Subgame-perfect equilibrium · 1965

He introduced the subgame-perfect equilibrium (subgame-perfect equilibrium, 1965) and the trembling-hand perfect equilibrium (trembling-hand perfect equilibrium, 1975).

Awards: Nobel Prize in Economics 1994 (with John Nash and John Harsanyi)

extensive form · backward induction · trembling hand

CHAPTERS **Chapter 2 · Games and Representations** **Chapter 3 · Nash Equilibrium**

Significance: These refinements eliminate Nash equilibria that rely on non-credible threats.

A subgame-perfect equilibrium forms a Nash equilibrium in every subgame. In finite games of perfect information it is found by backward induction. Selten also posed the chain-store paradox (1978); he is among the pioneers of experimental economics.

RA

Robert Aumann

b. 1930 · Israeli-American · Hebrew University of Jerusalem

Correlated equilibrium · repeated games

He introduced the correlated equilibrium (correlated equilibrium) (1974). He defined common knowledge formally ("Agreeing to Disagree", 1976) and developed the theory of repeated games.

Awards: Nobel Prize in Economics 2005 (with Thomas Schelling)

correlated equilibrium · common knowledge · folk theorem

CHAPTERS **Chapter 2 · Games and Representations** **Chapter 3 · Nash Equilibrium**

Significance: Correlated equilibria form a polyhedron that can be computed by linear programming. In learning that drives every player's internal regret to zero, the time-average distribution of the profiles played approaches the set of these equilibria.

In a correlated equilibrium, a mediator draws an action profile and privately recommends to each player their own action. No player gains by not following the recommendation. The folk theorem explains cooperation in infinitely repeated games.

KT

Daniel Kahneman & Amos Tversky

1934–2024 · 1937–1996 · Israeli-American · Princeton (Kahneman); Stanford (Tversky)

Prospect theory · 1979

They documented systematic biases in judgment (heuristics and biases, 1974). They proposed prospect theory (prospect theory) (Econometrica, 1979).

Awards: Kahneman: Nobel Prize in Economics 2002 (with Vernon Smith); Tversky died in 1996, before the prize

loss aversion · framing · probability weighting

CHAPTERS **Chapter 12 · Behavioral and Collective Decisions**

Significance: Prospect theory replaces expected utility with reference dependence, loss aversion and probability weighting. Behavioral game theory builds on this foundation.

Framing effects break the invariance assumed by rational choice models. Kahneman's book *Thinking, Fast and Slow* (2011) introduced the dual-process view to a wide audience.

RW

Robert Wilson

b. 1937 · American · Stanford University

Common-value auctions · auction formats

He pioneered the analysis of common-value auctions and the winner's curse (winner's curse) (1969, 1977). He is one of the designers of the simultaneous multiple-round auction.

Awards: Nobel Prize in Economics 2020 (with Paul Milgrom)

common value · winner's curse · sequential equilibrium

CHAPTERS **Chapter 2 · Games and Representations** **Chapter 5 · Auctions**

Significance: With Kreps he defined the sequential equilibrium (sequential equilibrium) (1982). With Milgrom he designed the spectrum auctions that the US Federal Communications Commission (FCC) has used since 1994.

Wilson was the doctoral advisor of Milgrom and also advised Roth and Rosenthal. His book *Nonlinear Pricing* (1993) influenced the design of electricity markets.

DB

Dietrich Braess

b. 1938 · German · Ruhr-Universität Bochum

Braess's paradox · 1968

In 1968 he showed that adding a new road to a network can increase every driver's travel time at equilibrium.

network paradox · Wardrop equilibrium · selfish routing

CHAPTERS **Chapter 8 · Network Design and Routing** **Chapter 9 · The Price of Anarchy**

Significance: Braess's paradox is the standard example showing that added capacity can harm selfish routing. With linear delays, the price of anarchy of this example is 4/3.

The paper ("Über ein Paradoxon aus der Verkehrsplanung", Unternehmensforschung, 1968) was translated into English in 2005. Braess's main field is numerical analysis (finite elements).

DK

Donald E. Knuth

b. 1938 · American · Stanford University

Analysis of alpha-beta pruning · 1975

In the paper "An analysis of alpha-beta pruning" (*Artificial Intelligence*, 1975), written with Ronald W. Moore, he measured how much work alpha-beta pruning saves.

Awards: Turing Award 1974

alpha-beta pruning | move ordering | effective branching factor

CHAPTERS **Chapter 2 · Minimax and Alpha-Beta Pruning**

Significance: When moves are examined in the best order, alpha-beta looks at $b^{\lceil d/2 \rceil} + b^{\lfloor d/2 \rfloor} - 1$ leaves in a uniform tree of branching factor b and depth d. This result is why roughly twice the depth can be searched in the same time.

The alpha-beta idea had been found independently by several researchers in the 1950s and 1960s. Knuth and Moore wrote the method down precisely and analyzed it. Knuth is better known for *The Art of Computer Programming* series and for TeX.

CG

Edward H. Clarke & Theodore Groves

Clarke 1939–2013 · American · Clarke: economist in the US federal government; Groves: UC San Diego

Clarke pivot rule 1971 · Groves mechanisms 1973

They generalized Vickrey's idea into the VCG mechanism. Clarke (Public Choice, 1971) introduced the pivot payment, and Groves (Econometrica, 1973) introduced the general class of mechanisms.

VCG | externality pricing | public good

CHAPTERS **Chapter 4 · Mechanism Design**

Significance: VCG (Vickrey 1961, Clarke 1971, Groves 1973) chooses the outcome that maximizes social welfare (social welfare). Each agent pays the externality it imposes on the others. Truthful reporting therefore becomes a dominant strategy.

Green and Laffont (1977) showed that under unrestricted quasilinear preferences the only mechanisms that are efficient in dominant strategies are the Groves mechanisms. VCG can run a budget deficit and is vulnerable to collusion.

AG

Allan Gibbard

b. 1942 · American · University of Michigan

Gibbard–Satterthwaite theorem · 1973

He proved that every voting rule with at least three possible outcomes that is not a dictatorship can be manipulated (Econometrica, 1973).

strategy-proofness | manipulation | dictatorship

CHAPTERS Chapter 4 · Mechanism Design | Chapter 7 · Social Choice and Voting

Significance: Gibbard and Satterthwaite proved the theorem independently of each other. The theorem makes strategy-proofness a fundamental constraint in mechanism design.

The ways out are these: restricted preference domains (single-peaked preferences and the median rule), money (VCG), randomization, and computational hardness of manipulation.

RA

Robert Axelrod

b. 1943 · American · University of Michigan

Repeated Prisoner's Dilemma tournaments · 1980

He organized computer tournaments for the repeated Prisoner's Dilemma (1980). Anatol Rapoport's Tit for Tat strategy won the tournaments.

Awards: MacArthur Fellowship 1987

repeated Prisoner's Dilemma | Tit for Tat | reciprocity

CHAPTERS Chapter 10 · Multi-Agent Systems | Chapter 13 · Learning and Adaptation

Significance: "The Evolution of Cooperation" (with W. D. Hamilton, Science 1981; book 1984) explained how reciprocity sustains cooperation.

Successful strategies were nice, retaliatory, forgiving and clear. The results inspired agent-based modeling and evolutionary dynamics.

MS

Mark Satterthwaite

b. 1945 · American · Kellogg School, Northwestern University

Gibbard–Satterthwaite 1975 · Myerson–Satterthwaite 1983

He proved the manipulability theorem independently of Gibbard (Journal of Economic Theory, 1975).

strategy-proofness | two-sided trade | impossibility

CHAPTERS Chapter 4 · Mechanism Design | Chapter 5 · Auctions | Chapter 7 · Social Choice and Voting

Significance: The Myerson–Satterthwaite theorem (1983) shows that in two-sided trade there is no mechanism that is at once efficient, incentive compatible, individually rational and budget balanced.

These two impossibility results explain why real markets and elections must give up some desirable properties.

RR

Robert W. Rosenthal

1945–2002 · American · Boston University

Congestion games · potential · 1973

He introduced congestion games (congestion games) (1973). With the help of a potential function he proved that these games always have a pure Nash equilibrium.

congestion game | potential function | pure equilibrium

CHAPTERS **Chapter 2 · Games and Representations** **Chapter 8 · Network Design and Routing** **Chapter 9 · The Price of Anarchy**

Significance: The Rosenthal potential guarantees that best-response dynamics terminate. It is the basic tool for routing games and for Price of Stability bounds.

Monderer and Shapley (1996) showed that every exact potential game is isomorphic to a congestion game. Rosenthal also introduced the centipede game (1981).

HV

Hal Varian

b. 1947 · American · UC Berkeley, Google

Position auctions · 2007

As Google's chief economist he studied advertising auctions in sponsored search (“Position Auctions”, 2007).

sponsored search | GSP auction | click-through rate

CHAPTERS **Chapter 6 · Online Platforms**

Significance: The symmetric Nash equilibria of the generalized second-price (generalized second-price, GSP) auction link ad prices to click-through rates and bidders' values.

Varian joined Google in 2002 and contributed to the design of the company's ad auction and reserve price policies. His textbook *Intermediate Microeconomics* is widely taught.

PM

Paul Milgrom

b. 1948 · American · Stanford University

Auction theory · market design

He developed the theory of affiliated values (affiliated values) and the linkage principle (linkage principle) (with Weber, 1982). He designed spectrum auctions.

Awards: Nobel Prize in Economics 2020 (with Robert Wilson)

linkage principle | ascending auction | spectrum auction

CHAPTERS **Chapter 5 · Auctions**

Significance: He is among the designers of the FCC's simultaneous multiple-round auction (1994) and of the 2016–17 incentive auction.

The linkage principle explains why open ascending auctions raise more revenue when values are affiliated. Book: *Putting Auction Theory to Work* (2004).

CP

Christos Papadimitriou

b. 1949 · Greek · UC Berkeley, Columbia University

PPAD 1994 · Price of Anarchy 1999

He defined the complexity class PPAD (1994). With Koutsoupias he introduced the ratio comparing the cost of the worst equilibrium with the optimum (1999); this ratio was later named the Price of Anarchy.

Awards: Knuth Prize 2002; Gödel Prize 2012 (for the Price of Anarchy work, with Koutsoupias); IEEE John von Neumann Medal 2016

PPAD | Price of Anarchy | total search problems

CHAPTERS **Chapter 2 · Games and Representations** **Chapter 3 · Nash Equilibrium** **Chapter 8 · Network Design and Routing** **Chapter 9 · The Price of Anarchy**

Significance: With Daskalakis and Goldberg he proved that computing a Nash equilibrium is PPAD-complete (2006).

PPAD covers total search problems whose solutions are guaranteed to exist by a parity argument on directed graphs (end-of-the-line, end-of-the-line). His 2001 STOC talk titled “Algorithms, Games, and the Internet” contributed to the birth of algorithmic game theory.

SH

Sergiu Hart

b. 1949 · Israeli · Hebrew University of Jerusalem

Regret matching · 2000

With Andreu Mas-Colell he introduced regret matching (regret matching) (Econometrica, 2000). In this simple adaptive method the observed play converges to the set of correlated equilibria.

regret matching | internal regret | correlated equilibrium

CHAPTERS **Chapter 13 · Learning and Adaptation**

Significance: Learning that drives internal regret to zero leads to correlated equilibria, and learning that drives external regret to zero leads to coarse correlated equilibria.

Hart and Mas-Colell also showed that "uncoupled" (uncoupled) dynamics cannot guarantee convergence to a Nash equilibrium in every game (2003).

EM

Eric Maskin

b. 1950 · American · Harvard University

Implementation theory · Maskin monotonicity · 1977

He determined which social choice rules can be implemented in Nash equilibrium (1977; published 1999).

Awards: Nobel Prize in Economics 2007 (with Leonid Hurwicz and Roger Myerson)

Nash implementation | monotonicity | social choice rules

CHAPTERS **Chapter 4 · Mechanism Design**

Significance: Maskin monotonicity is necessary for Nash implementation. It is also sufficient when there are three or more agents and no agent has veto power.

Implementation theory (implementation theory) asks whether all the equilibria of the designed game, not just some of them, give the desired outcome.

AR

Alvin Roth

b. 1951 · American · Harvard University, Stanford University

Market design · kidney exchange

He carried matching theory into practice: the redesign of the US medical residency match (Roth–Peranson, 1999), school choice in Boston and New York, and kidney exchange.

Awards: Nobel Prize in Economics 2012 (with Lloyd Shapley)

deferred acceptance · kidney exchange · school choice

CHAPTERS **Chapter 4 · Mechanism Design**

Significance: Market design (market design) treats markets as engineered systems. A good market must be thick, uncongested and make honest participation safe.

Roth showed that no stable matching mechanism is strategy-proof for both sides at once (1982). Book: *Who Gets What — and Why* (2015).

RM

Roger Myerson

b. 1951 · American · Northwestern University, University of Chicago

Revelation principle 1979 · optimal auctions 1981

He formulated the revelation principle (revelation principle) (1979). He characterized revenue-maximizing auctions (“Optimal Auction Design”, 1981).

Awards: Nobel Prize in Economics 2007 (with Leonid Hurwicz and Eric Maskin)

revelation principle · virtual value · Myerson's lemma

CHAPTERS **Chapter 4 · Mechanism Design** **Chapter 5 · Auctions**

Significance: Virtual values (virtual values) reduce revenue maximization to welfare maximization. When values are independent, identically distributed and regular, the optimal auction is a second-price auction with a reserve price.

Myerson's lemma: a single-parameter allocation rule is implementable if and only if it is monotone, and in that case the payment rule is uniquely determined. With Satterthwaite he proved the impossibility theorem for two-sided trade (1983).

TR

Jean Tirole & Jean-Charles Rochet

Tirole b. 1953 · French · Toulouse School of Economics

Two-sided markets · 2003

They founded the economics of two-sided platforms ("Platform Competition in Two-Sided Markets", 2003).

Awards: Tirole: Nobel Prize in Economics 2014 (alone)

two-sided market | network effect | platform pricing

CHAPTERS **Chapter 6 · Online Platforms**

Significance: Platforms set prices for both sides and may subsidize one side to attract the other. The pricing of app stores, card networks and marketplaces is explained by this framework.

Tirole received the Nobel Prize for his analyses of market power and regulation. With Drew Fudenberg he wrote the textbook *Game Theory* (1991).

YS

Yoav Shoham

Israeli-American · Stanford University

Multi-agent systems

He brought artificial intelligence and game theory together. With Kevin Leyton-Brown he wrote *Multiagent Systems: Algorithmic, Game-Theoretic, and Logical Foundations* (2009).

multi-agent system | agent design | computational equilibrium

CHAPTERS **Chapter 2 · Games and Representations** **Chapter 4 · Mechanism Design** **Chapter 10 · Multi-Agent Systems**

Significance: He carried strategic reasoning, negotiation and learning into computational models of autonomous agents.

He is among the founders of the AI Index report and of the company AI21 Labs.

ÉT

Éva Tardos

b. 1957 · Hungarian-American · Cornell University

Selfish routing · 2000–02

With Roughgarden she bounded the Price of Anarchy in selfish routing (selfish routing) ("How Bad Is Selfish Routing?", FOCS 2000; JACM 2002).

Awards: Fulkerson Prize 1988; Gödel Prize 2012; IEEE John von Neumann Medal 2019

network flows | Price of Anarchy | learning in games

CHAPTERS **Chapter 2 · Games and Representations**

Chapter 8 · Network Design and Routing | Chapter 9 · The Price of Anarchy

Significance: With linear delays, the cost of a Wardrop equilibrium is at most 4/3 times the optimum, whatever the network.

She is one of the editors of the book *Algorithmic Game Theory* (2007). With Kleinberg she wrote *Algorithm Design* (2005). Her recent work studies players who learn without regret in auctions and routing.

VV

Vijay V. Vazirani

b. 1957 · Indian-born American · Georgia Tech, UC Irvine

AdWords · online matching · market equilibria

He co-developed the AdWords algorithm (Mehta, Saberi, U. Vazirani, V. Vazirani, 2005). The algorithm achieves a competitive ratio (competitive ratio) of 1−1/e in budgeted online allocation.

online matching | AdWords | market equilibrium

CHAPTERS Chapter 2 · Games and Representations | Chapter 6 · Online Platforms

Significance: Online bipartite matching (Karp–Vazirani–Vazirani, 1990) and its AdWords generalization model how ad platforms assign queries to advertisers.

With Nisan, Roughgarden and Tardos he edited the book *Algorithmic Game Theory* (2007). He also designed combinatorial algorithms for Fisher market equilibria.

JS

Jonathan Schaeffer

b. 1957 · Canadian · University of Alberta

Chinook · solving checkers · 2007

He developed the checkers program Chinook. Chinook was declared man–machine world champion in 1994. His team solved checkers in 2007: if neither side makes a mistake, the game ends in a draw (*Science*, 2007).

solved games | checkers | endgame database

CHAPTERS Chapter 2 · Games and Representations

Significance: By Zermelo's theorem the outcome of checkers under best play is determined. Checkers is among the largest games for which this outcome has actually been found by computer.

The solution is a weak solution: it gives the value of the starting position and a line of play that preserves this value, but not the best move for every position. The search combined alpha-beta variants with very large endgame databases.

CD

Cynthia Dwork

b. 1958 · American · Harvard University

Differential privacy 2006 · fairness through awareness 2012

She co-invented differential privacy (differential privacy) (Dwork, McSherry, Nissim, Smith, 2006). She defined individual fairness (individual fairness) ("Fairness Through Awareness", 2012).

Awards: Gödel Prize 2017; Knuth Prize 2020

differential privacy · individual fairness · algorithmic ethics

CHAPTERS **Chapter 7 · Social Choice and Voting** · **Chapter 11 · Algorithmic Fairness and Ethics** · **Chapter 14 · Future Directions**

Significance: Individual fairness demands that similar individuals be treated similarly. Approximately truthful mechanisms also follow from differential privacy (McSherry–Talwar, 2007).

Her work treats privacy and fairness as mathematical guarantees that can trade off against accuracy. Her work on rank aggregation methods for the web (with Kumar, Naor and Sivakumar, 2001) is connected to voting theory.

DB

Feng-hsiung Hsu, Murray Campbell & A. Joseph Hoane

The Deep Blue team · Carnegie Mellon University, IBM Research

Deep Blue · 1997

The programs ChipTest and Deep Thought, which began with Hsu's doctoral work at Carnegie Mellon, became Deep Blue at IBM. In 1997 Deep Blue beat world champion Garry Kasparov 3½–2½ in a six-game match.

Awards: Fredkin Prize 1997 (the Deep Blue team)

parallel alpha-beta · chess chip · evaluation function

CHAPTERS **Chapter 2 · Minimax and Alpha-Beta Pruning**

Significance: Deep Blue showed that depth-limited alpha-beta search with special-purpose hardware can beat a world champion. The 1997 version evaluated about 200 million positions per second.

Kasparov had won the first match in 1996 by 4–2. Hsu (b. 1959) is the chief designer of the system and told the story in the book *Behind Deep Blue* (2002). He described the system in the paper "Deep Blue" (*Artificial Intelligence*, 2002), written with the other two.

NN

Noam Nisan

b. 1961 · Israeli · Hebrew University of Jerusalem

Algorithmic mechanism design · 1999

With Amir Ronen he founded algorithmic mechanism design ("Algorithmic Mechanism Design", STOC 1999; GEB 2001).

Awards: Gödel Prize 2012 (with Amir Ronen); Knuth Prize 2016

truthful approximation algorithms · combinatorial auction · communication complexity

CHAPTERS **Chapter 2 · Games and Representations** **Chapter 4 · Mechanism Design** **Chapter 6 · Online Platforms**

Significance: Mechanisms must be both incentive compatible and computationally efficient. VCG run with an approximation algorithm is generally not truthful.

He is one of the editors of the book *Algorithmic Game Theory* (2007). His work on combinatorial auctions measures how much communication is needed for an efficient allocation.

MK

Michael Kearns

American · University of Pennsylvania

Graphical games · ethical algorithms

He introduced graphical games (with Littman and Singh, 2001). He studied fairness and privacy in machine learning.

graphical games · learning theory · algorithmic fairness

CHAPTERS **Chapter 11 · Algorithmic Fairness and Ethics** **Chapter 14 · Future Directions**

Significance: When each player's payoff depends only on their neighbors, graphical games represent large games compactly.

With Aaron Roth he wrote *The Ethical Algorithm* (2019), which describes how to build fairness and privacy into algorithms.

EK

Elias Koutsoupias

Greek · University of Oxford

Worst-case equilibria · 1999

With Papadimitriou he introduced the ratio between the worst equilibrium and the optimum ("Worst-case Equilibria", STACS 1999). This ratio was later called the Price of Anarchy.

Awards: Gödel Prize 2012 (with Christos Papadimitriou)

Price of Anarchy · load balancing · worst-case analysis

CHAPTERS **Chapter 9 · The Price of Anarchy**

Significance: The Price of Anarchy gives a guarantee on how much selfish behavior can degrade system performance in the worst case.

The first model was load balancing of selfish jobs over parallel links. Koutsoupias also worked on online algorithms (the k-server conjecture for the work function algorithm).

PG

Paul W. Goldberg

British · University of Oxford

Nash equilibrium is PPAD-complete · 2006

With Daskalakis and Papadimitriou he proved the PPAD-completeness of Nash equilibrium (2006).

PPAD · graphical games · reductions

CHAPTERS **Chapter 3 · Nash Equilibrium**

Significance: The reduction encodes arithmetic circuits as games. Every equilibrium then solves an end-of-the-line instance.

Goldberg also studies the complexity of approximate equilibria and of path-following methods such as Lemke–Howson.

AA

James D. Allen & Victor Allis

Allis: Dutch, b. 1965 · Vrije Universiteit Amsterdam, Maastricht University

Solving Connect Four · 1988

They solved Connect Four (Connect Four) independently in 1988: if the first player starts in the center column, they are guaranteed a win. Allen's result was announced on 1 October and Allis's on 16 October.

solved games · knowledge-based search · proof-number search

CHAPTERS Chapter 2 · Games and Representations

Chapter 2 · Minimax and Alpha-Beta Pruning

Significance: The game has about $4.5 \cdot 10^{12}$ legal positions. The solution was obtained not by brute force but with game-specific rules and powerful search.

In his master's thesis Allis used a knowledge-based approach built on nine rules. In his 1994 doctoral thesis he introduced proof-number search (proof-number search).

JK

Jon Kleinberg

b. 1971 · American · Cornell University

Networks · fairness trade-offs · 2016

With Mullainathan and Raghavan he proved that calibration and balanced error rates in risk scores cannot hold together except in special cases (2016).

Awards: Nevanlinna Prize 2006

fairness impossibility · networks · HITS

CHAPTERS Chapter 6 · Online Platforms · Chapter 11 · Algorithmic Fairness and Ethics

Significance: This impossibility result shows that fairness criteria conflict with one another. Algorithmic fairness therefore requires making explicit choices.

The HITS algorithm ranks web pages as hubs (hubs) and authorities (authorities). His book *Networks, Crowds, and Markets* (2010), written with Easley, also covers sponsored search, matching markets and Braess's paradox.

TR

Tim Roughgarden

b. 1975 · American · Stanford University, Columbia University

Selfish routing · smoothness

With Tardos he proved tight bounds on the Price of Anarchy in selfish routing (2000–02). Later he introduced smooth games (smooth games) (2009).

Awards: Gödel Prize 2012 (with Éva Tardos)

Price of Anarchy | smoothness | selfish routing

CHAPTERS **Chapter 2 · Games and Representations** **Chapter 3 · Nash Equilibrium** **Chapter 4 · Mechanism Design** **Chapter 5 · Auctions** **Chapter 8 · Network Design and Routing** **Chapter 9 · The Price of Anarchy** **Chapter 13 · Learning and Adaptation** **Chapter 14 · Future Directions**

Significance: Smoothness arguments extend Price of Anarchy bounds to mixed, correlated and no-regret outcomes.

His book *Twenty Lectures on Algorithmic Game Theory* (2016) is a widely used course resource. His recent work studies blockchain transaction fee mechanisms.

JH

Jason Hartline

American · Northwestern University

Mechanism design and approximation

He developed simple prior-independent (prior-independent) mechanisms that approximately achieve optimal revenue.

prior-independent mechanisms | approximation | revenue

CHAPTERS **Chapter 4 · Mechanism Design** **Chapter 5 · Auctions**

Significance: He shows when simple auctions (for example, a second-price auction with an anonymous reserve price) are approximately optimal. In this way he bridges Myerson's theory and practice.

His book *Mechanism Design and Approximation* develops Bayesian mechanism design from an algorithmic point of view.

DS

David Silver & Demis Hassabis

b. 1976 · b. 1976 · Google DeepMind

AlphaGo · 2016

The AlphaGo project, led by Silver, combined Monte Carlo tree search with deep neural networks. AlphaGo beat Lee Sedol 4–1 in March 2016 (*Nature*, 2016). Hassabis co-founded DeepMind in 2010.

Awards: Silver: ACM Prize in Computing 2019 · Hassabis: Nobel Prize in Chemistry 2024 (with John M. Jumper, for protein structure prediction)

Monte Carlo tree search | neural-network evaluation | self-play

CHAPTERS Chapter 2 · Minimax and Alpha-Beta Pruning

Chapter 13 · Learning and Adaptation

Significance: In Go the branching factor is about 250, and designing a good evaluation function by hand is very hard. AlphaGo left evaluation and move selection to learned networks.

The AlphaZero that followed learned Go, chess and shogi without human games, purely by self-play. The search is not minimax: it uses Monte Carlo tree search guided by learned policy and value networks. The idea shared with minimax is to look ahead assuming the opponent's good reply.

AP

Ariel Procaccia

Israeli-American · Carnegie Mellon University, Harvard University

Computational social choice · fair division

He applied algorithms to voting and fair division. He is among the creators of the website Spliddit, built for provably fair division of rent, shared work contributions and tasks.

Awards: IJCAI Computers and Thought Award 2015

envy-freeness | computational social choice | citizens' assemblies

CHAPTERS Chapter 7 · Social Choice and Voting

Chapter 11 · Algorithmic Fairness and Ethics

Significance: Envy-freeness up to one good (envy-free up to one good, EF1), maximum Nash social welfare and leximin allocations make fairness guarantees practical.

He also designed algorithms for selecting representative citizens' assemblies by lottery.

CD

Constantinos Daskalakis

b. 1981 · Greek · MIT

Nash equilibrium is PPAD-complete · 2006

With Goldberg and Papadimitriou he proved that computing a Nash equilibrium is PPAD-complete (STOC 2006; SIAM J. Computing 2009).

Awards: Nevanlinna Prize 2018

PPAD-completeness | equilibrium computation | min-max optimization

CHAPTERS Chapter 2 · Games and Representations | Chapter 3 · Nash Equilibrium

Chapter 14 · Future Directions

Significance: Unless PPAD is easy, there is no efficient algorithm that finds a Nash equilibrium in general games. This result calls into question using equilibrium as a prediction.

The DGP proof covers games with three or more players. Chen and Deng (2006) extended the hardness to two-player games. His recent work studies learning dynamics in min-max optimization and multi-agent learning.

EOS

Benjamin Edelman, Michael Ostrovsky & Michael Schwarz

Harvard, Stanford and Yahoo! Research (their institutions in 2007)

Generalized second-price auction · 2007

They studied the generalized second-price (GSP) auction used in search advertising (American Economic Review, 2007).

GSP · sponsored search · envy-free equilibrium

CHAPTERS Chapter 6 · Online Platforms

Significance: GSP is not truthful. Even so, its lowest-revenue locally envy-free equilibrium gives the same placements and payments as VCG.

Independently of Varian, they explained why GSP prices remain stable in practice even though truthful bidding is not a dominant strategy.

MC

Rémi Coulom, Levente Kocsis & Csaba Szepesvári

Coulom: French · Kocsis and Szepesvári: MTA SZTAKI, Budapest (their institutions in 2006)

Monte Carlo tree search and UCT · 2006

In 2006 Coulom combined game tree search with randomly played games and named the method Monte Carlo tree search (MCTS). In the same year Kocsis and Szepesvári proposed UCT, which applies the bandit algorithm UCB1 to the tree (ECML 2006).

MCTS · UCT · exploration and exploitation

CHAPTERS Chapter 2 · Minimax and Alpha-Beta Pruning

Chapter 13 · Learning and Adaptation

Significance: In games where a good evaluation function cannot be written, MCTS evaluates positions by the results of random playouts. It is the search skeleton of strong Go programs and of AlphaGo.

Coulom's program Crazy Stone won the 9×9 Go tournament at the 2006 Computer Olympiad. UCT treats move selection at each node as a multi-armed bandit problem.

Award years show the year in which the prize was given. “Nobel Prize in Economics” stands for the Sveriges Riksbank Prize in Economic Sciences in Memory of Alfred Nobel. People whose results are described together share a card. On cards without a birth year, the order follows the period of the main work.