

Bölüm 14: Gelecek Yönelimler

Algoritmik Oyun Kuramı

Yapay zekâ ajanları

Öğrenen mekanizmalar

Blokzincir mekanizmaları

Hesaplama ve ölçek

Dersin haritası

Proje yönergesi

Sercan KÜLCÜ · Giresun Üniversitesi · Bilgisayar Mühendisliği

Öğrenme Çıktıları

Yeni oyuncular

Yapay zekâ ajanlarının pazarlık ve fiyatlandırmada yarattığı oyun kuramı sorularını tanımlamak.

Veriden tasarım

Mekanizmanın veriden öğrenilmesini ve örneklem karmaşıklığı kavramını sayısal örneklerle açıklamak.

Zincir üstü mekanizmalar

İşlem ücreti açık artırmalarını, taban ücret kuralını ve MEV kavramını yorumlamak.

Dersi birleştirmek

Dersin 14 bölümünü tek haritada ilişkilendirmek ve bir dönem projesi tasarlamak.

Algoritmik Oyun Kuramı: Gelişen Bir Alan

- Endüstri ve akademide uygulamalar artıyor: otonom araçlar, çevrimiçi platformlar, enerji ağları.
- Teknolojik eğilimler: yapay zekâ ve makine öğrenmesi, büyük veri ve gerçek zamanlı karar, blokzincir ve akıllı sözleşmeler.

Adalet ve etik

Adil ve şeffaf mekanizmalar (Bölüm 10).

Dinamik ve öğrenen sistemler

Uyarlanır otonom ajanlar (Bölüm 12).

Büyük ağlar

Milyonlarca oyunculu stratejik etkileşim (Bölüm 8, 11).

Teşvik ve politika

Sosyal refah ve sürdürülebilirlik için kural tasarımı (Bölüm 4).

Bu bölüm, dersin araçlarının bu yeni alanlarda nasıl kullanıldığını ve nerede yetersiz kaldığını gösterir.

Uygulama Alanları

Finans

Algoritmik ticaret, portföy yönetimi; alım-satım algoritmaları birbirine karşı oynar.

Trafik ve lojistik

Otonom araç ağları, rota seçimi; bencil yönlendirme ve Braess (Bölüm 8).

Çevrimiçi platformlar

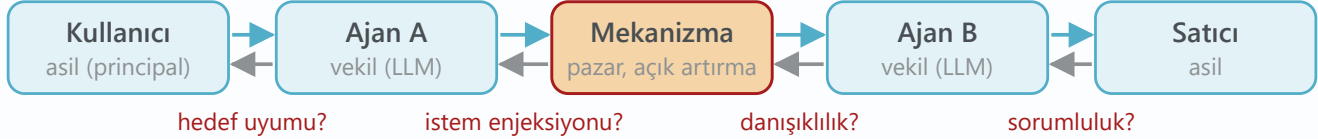
Öneri sistemleri, reklam açık artırmaları, sosyal ağ analizi (Bölüm 9).

Enerji şebekeleri

Yük dengeleme, talep yanıtı, elektrik piyasaları (Bölüm 5, 11).

Stratejik Yapay Zekâ Ajanları

- Büyük dil modeli (LLM) tabanlı ajanlar kullanıcı adına alışveriş, pazarlık, planlama ve kaynak paylaşımı giderek daha çok üstleniyor.
- Ajan artık bir **oyuncu**: hedefi, bilgisi ve stratejisi olan bir aktör. Oyun kuramının soruları yeniden gündemde.



Hedef uyumu

Ajanın hedefi kullanıcıyla aynı mı?
Asil–vekil (principal–agent) sorunu.

Rasyonellik varsayımı

LLM davranışı çerçevelemeye (framing) duyarlı olabilir; Nash tahmininden sapabilir.

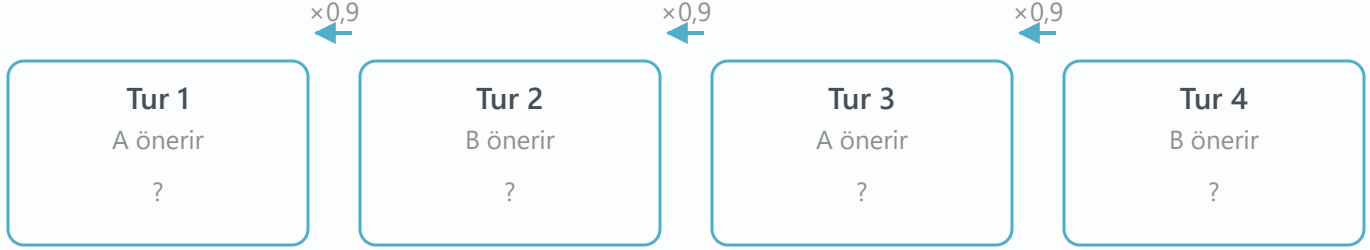
Manipülasyon

Karşı ajanın mesajına gizlenen talimatlar (istem enjeksiyonu: prompt injection).

Kimlik ve sorumluluk

Kim adına konuşuyor, hatadan kim sorumlu, sözleşme nasıl doğrulanır?

Adım Adım: Ajanlar Arası Pazarlık



son tur: reddedilirse ikisi de 0 alır

Tur sayısı T	1	2	3	4	5	6	7	8	∞
A'nın payı									

100 TL paylaşılacak. A tek, B çift turlarda önerir; her gecikme değeri %10 azaltır ($\delta = 0,9$). Tur 4 son tur. Geriye doğru çıkarım (Bölüm 2).

Adım Adım: Ajanlar Arası Pazarlık

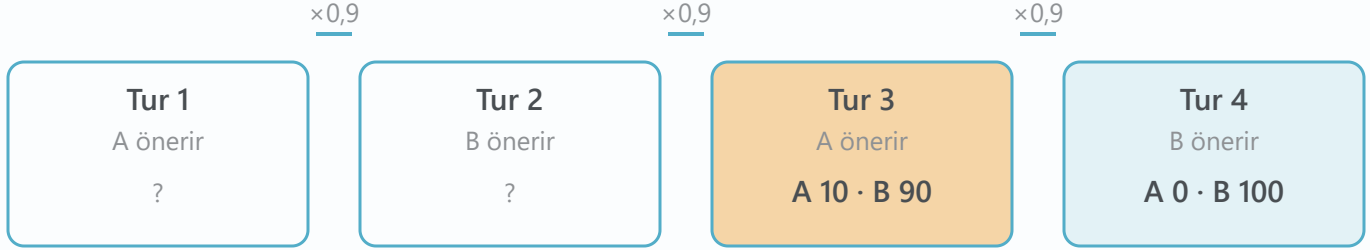


son tur: reddedilirse ikisi de 0 alır

Tur sayısı T	1	2	3	4	5	6	7	8	∞
A'nın payı									

Tur 4: B önerir. A reddederse ikisi de 0 alır $\rightarrow$ A her şeyi kabul eder: **B 100, A 0.**

Adım Adım: Ajanlar Arası Pazarlık

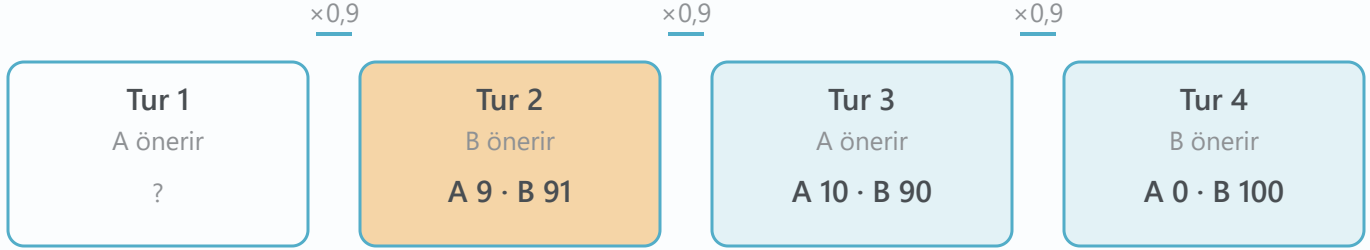


son tur: reddedilirse ikisi de 0 alır

Tur sayısı T	1	2	3	4	5	6	7	8	∞
A'nın payı									

Tur 3: B reddederse tur 4'te 100 alır; bugünkü değeri $0,9 \cdot 100 = 90$. A, B'ye 90 önerir: **A 10, B 90**.

Adım Adım: Ajanlar Arası Pazarlık

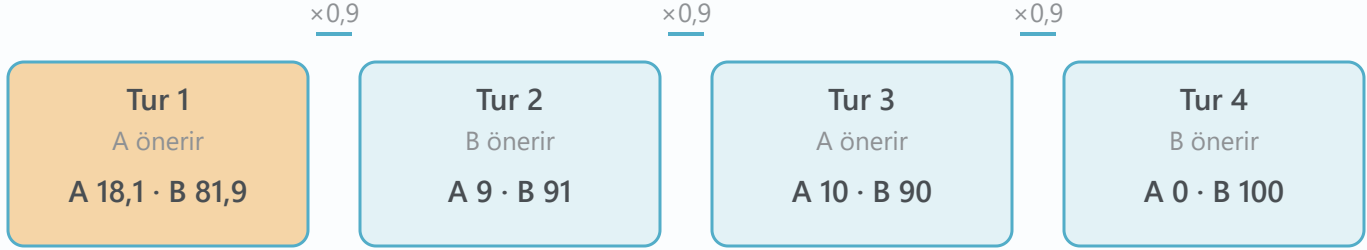


son tur: reddedilirse ikisi de 0 alır

Tur sayısı T	1	2	3	4	5	6	7	8	∞
A'nın payı									

Tur 2: A'nın bekleme değeri $0,9 \cdot 10 = 9$. B, A'ya 9 önerir: **A 9, B 91**.

Adım Adım: Ajanlar Arası Pazarlık

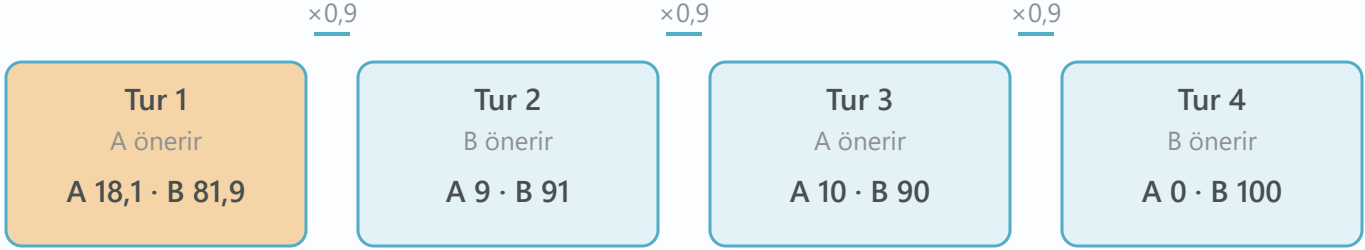


son tur: reddedilirse ikisi de 0 alır

Tur sayısı T	1	2	3	4	5	6	7	8	∞
A'nın payı									

Tur 1: B'nin bekleme değeri $0,9 \cdot 91 = 81,9$. A 18,1 alır, anlaşma **hemen** olur (alt oyun mükemmel denge).

Adım Adım: Ajanlar Arası Pazarlık



son tur: reddedilirse ikisi de 0 alır

Tur sayısı T	1	2	3	4	5	6	7	8	∞
A'nın payı	100	10	91	18,1	83,7	24,7	77,8	30	52,6

Tur sayısı değişince pay salınır; sonsuz ufukta (Rubinstein, 1982) önerenin payı $1/(1 + \delta) = \mathbf{52,6}$.

Adım Adım: Ajanlar Arası Pazarlık

×0,9

×0,9

×0,9

Tur 1

A önerir

A 18,1 · B 81,9

Tur 2

B önerir

A 9 · B 91

Tur 3

A önerir

A 10 · B 90

Tur 4

B önerir

A 0 · B 100

son tur: reddedilirse ikisi de 0 alır

Tur sayısı T	1	2	3	4	5	6	7	8	∞
A'nın payı	100	10	91	18,1	83,7	24,7	77,8	30	52,6

Ajanlar için ders: **sabır (δ) ve son söz hakkı pazarlık gücüdür**. Son tarih, son teklifin kimde olduğunu belirlediği için ufuk bilgisi de stratejiktir.

Fiyatlandırma Algoritmaları Danışabilir mi?

- İki firma aynı ürünü satıyor; tekel fiyatında toplam kâr 100 (her birine 50). Fiyat kırılırsa kâr sıfırlanır (Bertrand).
- Tetik (kinci, Grim) stratejisi: “Rakip fiyat kırana kadar yüksek fiyat, kırarsa sonsuza dek rekabet.” (Bölüm 12)
- Sapma: bir dönem 100, sonra 0. İşbirliği: her dönem 50 → toplam $50/(1 - \delta)$.

$$50/(1 - \delta) \geq 100 \Leftrightarrow \delta \geq 1/2$$

δ	işbirliği	sapma	sürer mi?
0,3	71,4	100	hayır
0,5	100	100	evet
0,7	166,7	100	evet
0,9	500	100	evet

Simülasyonlarda pekiştirmeli öğrenen fiyatlandırma ajanlarının, açıkça iletişim kurmadan rekabetçi düzeyin üstünde fiyatlar öğrenebildiği gösterilmiştir (Calvano ve diğ., 2020).

Düzenleme sorusu: açık bir anlaşma yoksa **algoritmik danışıklılık** (algorithmic collusion) var mıdır? Rekabet hukuku “insanlar arası anlaşma” üzerine kuruludur.

Otomatik Mekanizma Tasarımı

Klasik yol

- İnsan bir kural önerir (Vickrey, VCG, Myerson).
- Teşvik uyumluluğu ve optimalliği elle ispatlanır.
- Tek nesne gibi yapıli problemlerde çok başarılı; çok nesneli, çok alıcılı durumlarda optimum çoğu zaman bilinmez.

Otomatik yol

- Mekanizma bir optimizasyon probleminin çözümüdür: amaç (gelir, refah) + kısıtlar (IC, IR).
- Sonlu tip kümelerinde doğrusal programlama (Conitzer ve Sandholm, 2002).
- Sinir ağı ile açık artırma: IC kısıtı "pişmanlık" cezasına dönüşür (Dütting ve diğ., 2019).

Yaklaşık IC

Öğrenilen kural tam doğru olmayabilir; küçük pişmanlık kalır.

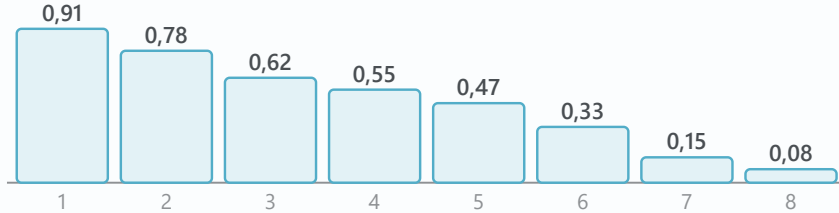
Genelleme

Eğitim verisindeki dağılım değişirse kural etkisizleşebilir.

Açıklanabilirlik

"Neden bu fiyat?" sorusu ağırlıklarla yanıtlanamaz.

Kod İzleme: Veriden Rezerv Fiyatı Öğrenmek



Geçmişte gözlenen 8 alıcı değeri büyükten küçüğe sıralanır.

Adım 1 / 12

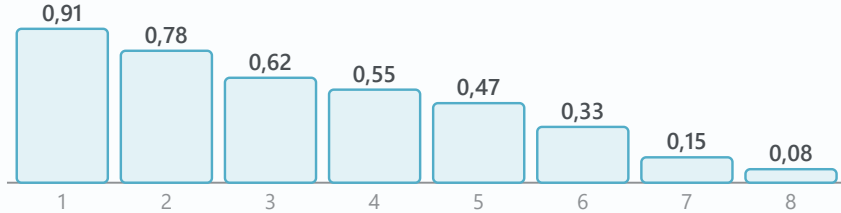
```
1 def en_yi_fiyat(orn):  
    ▶ v = sorted(orn, reverse=True)  
    3 n = len(v)  
    4 fiyat, gelir = 0, 0  
    5 for k in range(1, n + 1):  
    6     p = v[k - 1]          # k. en yüksek  
    7     g = p * k / n          # k alıcı alır  
    8     if g > gelir:  
    9         fiyat, gelir = p, g  
    10 return fiyat, gelir
```

DEĞİŞKENLER

orn [0.62, 0.15, 0.91, 0.47, 0.33,
= 0.78, 0.55, 0.08]

v [0.91, 0.78, 0.62, 0.55, 0.47,
= 0.33, 0.15, 0.08]

Kod İzleme: Veriden Rezerv Fiyatı Öğrenmek



$n = 8$ örnek. Fiyat p ise örneklerden p 'ye eşit veya büyük olanlar satın alır.

Adım 2 / 12

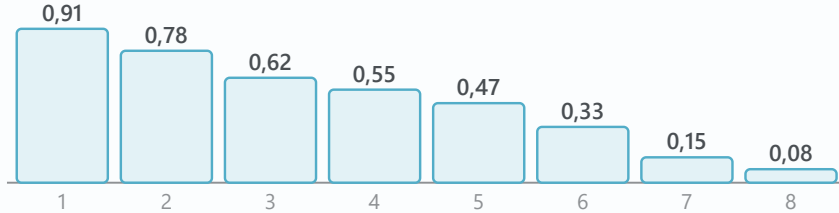
```
1 def en_iyi_fiyat(orn):
2     v = sorted(orn, reverse=True)
3     n = len(v)
4     fiyat, gelir = 0, 0
5     for k in range(1, n + 1):
6         p = v[k - 1]          # k. en yüksek
7         g = p * k / n          # k alıcı alır
8         if g > gelir:
9             fiyat, gelir = p, g
10    return fiyat, gelir
```

DEĞİŞKENLER

v [0.91, 0.78, 0.62, 0.55, 0.47, 0.33, 0.15, 0.08]

$n = 8$

Kod İzleme: Veriden Rezerv Fiyatı Öğrenmek



Şimdiye kadarki en iyi fiyat ve deneysel gelir.

Adım 3 / 12

```
1 def en_iyi_fiyat(orn):
2     v = sorted(orn, reverse=True)
3     n = len(v)
4     fiyat, gelir = 0, 0
5     for k in range(1, n + 1):
6         p = v[k - 1]          # k. en yüksek
7         g = p * k / n         # k alıcı alır
8         if g > gelir:
9             fiyat, gelir = p, g
10    return fiyat, gelir
```

DEĞİŞKENLER

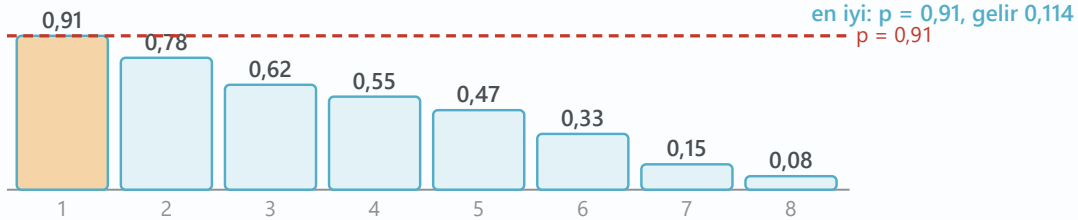
v [0.91, 0.78, 0.62, 0.55, 0.47,
= 0.33, 0.15, 0.08]

n = 8

fiyat = 0

gelir = 0

Kod İzleme: Veriden Rezerv Fiyatı Öğrenmek



$k = 1: p = 0,91 \rightarrow 1$ alıcı alır, $g = 0,91 \cdot 1/8 = \mathbf{0,114} \rightarrow$ yeni en iyi.

Adım 4 / 12

```

1 def en_yii_fiyat(orn):
2     v = sorted(orn, reverse=True)
3     n = len(v)
4     fiyat, gelir = 0, 0
5     for k in range(1, n + 1):
6         p = v[k - 1]          # k. en yüksek
7         g = p * k / n         # k alıcı alır
8         if g > gelir:
9             fiyat, gelir = p, g
10    return fiyat, gelir
  
```

DEĞİŞKENLER

v [0.91, 0.78, 0.62, 0.55, 0.47, 0.33, 0.15, 0.08]

$n = 8$

$k = 1$

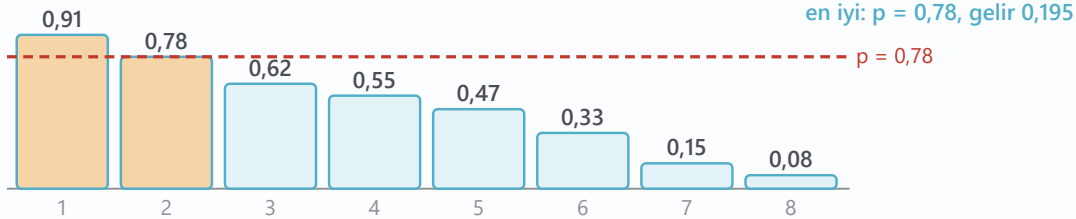
$p = 0.91$

$g = 0.114$

$\text{fiyat} = 0.91$

$\text{gelir} = 0.114$

Kod İzleme: Veriden Rezerv Fiyatı Öğrenmek



$k = 2$: $p = 0,78 \rightarrow 2$ alıcı alır, $g = 0,78 \cdot 2 / 8 = \mathbf{0,195} \rightarrow$ yeni en iyi.

Adım 5 / 12

```

1 def en_iyi_fiyat(orn):
2     v = sorted(orn, reverse=True)
3     n = len(v)
4     fiyat, gelir = 0, 0
5     for k in range(1, n + 1):
6         p = v[k - 1]          # k. en yüksek
7         g = p * k / n         # k alıcı alır
8         if g > gelir:
9             fiyat, gelir = p, g
10    return fiyat, gelir

```

DEĞİŞKENLER

v [0.91, 0.78, 0.62, 0.55, 0.47, 0.33, 0.15, 0.08]

$n = 8$

$k = 2$

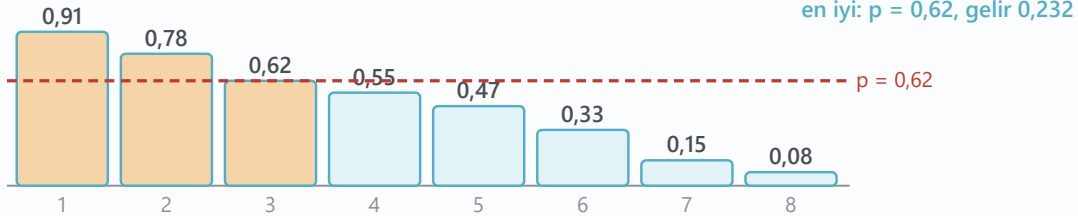
$p = 0.78$

$g = 0.195$

fiyat = 0.78

gelir = 0.195

Kod İzleme: Veriden Rezerv Fiyatı Öğrenmek



$k = 3$: $p = 0,62 \rightarrow 3$ alıcı alır, $g = 0,62 \cdot 3 / 8 = \mathbf{0,232} \rightarrow$ yeni en iyi.

Adım 6 / 12

```

1 def en_iyi_fiyat(orn):
2     v = sorted(orn, reverse=True)
3     n = len(v)
4     fiyat, gelir = 0, 0
5     for k in range(1, n + 1):
6         p = v[k - 1]          # k. en yüksek
7         g = p * k / n         # k alıcı alır
8         if g > gelir:
9             fiyat, gelir = p, g
10    return fiyat, gelir

```

DEĞİŞKENLER

v [0.91, 0.78, 0.62, 0.55, 0.47, 0.33, 0.15, 0.08]

$n = 8$

$k = 3$

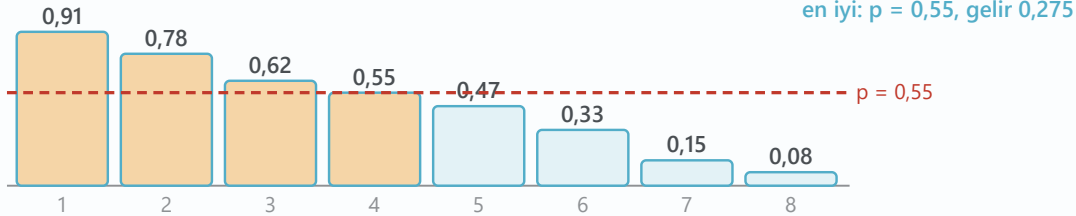
$p = 0.62$

$g = 0.232$

$\text{fiyat} = 0.62$

$\text{gelir} = 0.232$

Kod İzleme: Veriden Rezerv Fiyatı Öğrenmek



$k = 4$: $p = 0,55 \rightarrow 4$ alıcı alır, $g = 0,55 \cdot 4 / 8 = \mathbf{0,275} \rightarrow$ yeni en iyi.

Adım 7 / 12

```

1 def en_yii_fiyat(orn):
2     v = sorted(orn, reverse=True)
3     n = len(v)
4     fiyat, gelir = 0, 0
5     for k in range(1, n + 1):
6         p = v[k - 1]          # k. en yüksek
7         g = p * k / n         # k alıcı alır
8         if g > gelir:
9             fiyat, gelir = p, g
10    return fiyat, gelir

```

DEĞİŞKENLER

v [0.91, 0.78, 0.62, 0.55, 0.47,
= 0.33, 0.15, 0.08]

n = 8

k = 4

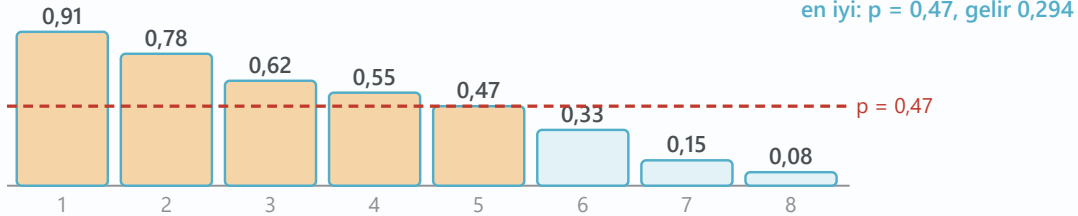
p = 0.55

g = 0.275

fiyat = 0.55

gelir = 0.275

Kod İzleme: Veriden Rezerv Fiyatı Öğrenmek



$k = 5$: $p = 0,47 \rightarrow 5$ alıcı alır, $g = 0,47 \cdot 5 / 8 = \mathbf{0,294} \rightarrow$ yeni en iyi.

Adım 8 / 12

```

1 def en_iyi_fiyat(orn):
2     v = sorted(orn, reverse=True)
3     n = len(v)
4     fiyat, gelir = 0, 0
5     for k in range(1, n + 1):
6         p = v[k - 1]          # k. en yüksek
7         g = p * k / n         # k alıcı alır
8         if g > gelir:
9             fiyat, gelir = p, g
10    return fiyat, gelir

```

DEĞİŞKENLER

v [0.91, 0.78, 0.62, 0.55, 0.47, 0.33, 0.15, 0.08]

$n = 8$

$k = 5$

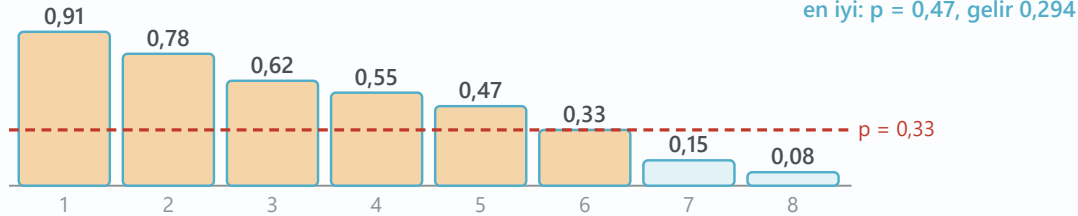
$p = 0.47$

$g = 0.294$

$\text{fiyat} = 0.47$

$\text{gelir} = 0.294$

Kod İzleme: Veriden Rezerv Fiyatı Öğrenmek



k = 6: p = 0,33 → 6 alıcı alır, g = 0,33·6/8 = **0,247** ≤ 0,294, değişmez.

Adım 9 / 12

```

1 def en_yii_fiyat(orn):
2     v = sorted(orn, reverse=True)
3     n = len(v)
4     fiyat, gelir = 0, 0
5     for k in range(1, n + 1):
6         p = v[k - 1]          # k. en yüksek
7         g = p * k / n         # k alıcı alır
8         if g > gelir:
9             fiyat, gelir = p, g
10    return fiyat, gelir
  
```

DEĞİŞKENLER

v [0.91, 0.78, 0.62, 0.55, 0.47,
= 0.33, 0.15, 0.08]

n = 8

k = 6

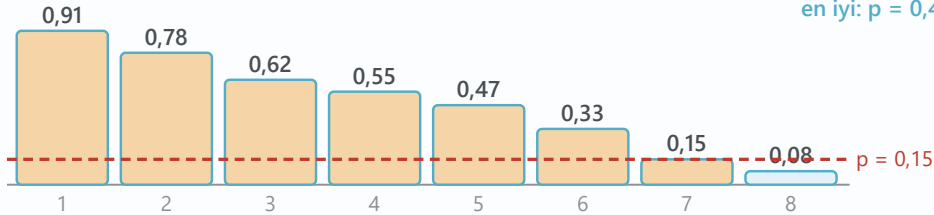
p = 0.33

g = 0.247

fiyat = 0.47

gelir = 0.294

Kod İzleme: Veriden Rezerv Fiyatı Öğrenmek



$k = 7$: $p = 0,15 \rightarrow 7$ alıcı alır, $g = 0,15 \cdot 7 / 8 = \mathbf{0,131} \leq 0,294$, değişmez.

Adım 10 / 12

```

1 def en_iyi_fiyat(orn):
2     v = sorted(orn, reverse=True)
3     n = len(v)
4     fiyat, gelir = 0, 0
5     for k in range(1, n + 1):
6         p = v[k - 1]          # k. en yüksek
7         g = p * k / n         # k alıcı alır
8         if g > gelir:
9             fiyat, gelir = p, g
10    return fiyat, gelir

```

DEĞİŞKENLER

v [0.91, 0.78, 0.62, 0.55, 0.47, 0.33, 0.15, 0.08]

$n = 8$

$k = 7$

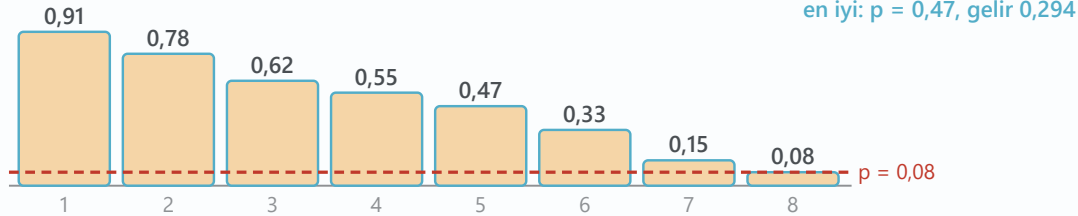
$p = 0.15$

$g = 0.131$

fiyat = 0.47

gelir = 0.294

Kod İzleme: Veriden Rezerv Fiyatı Öğrenmek



$k = 8$: $p = 0,08 \rightarrow 8$ alıcı alır, $g = 0,08 \cdot 8 / 8 = \mathbf{0,08} \leq 0,294$, değişmez.

Adım 11 / 12

```

1 def en_iyi_fiyat(orn):
2     v = sorted(orn, reverse=True)
3     n = len(v)
4     fiyat, gelir = 0, 0
5     for k in range(1, n + 1):
6         p = v[k - 1]          # k. en yüksek
7         g = p * k / n          # k alıcı alır
8         if g > gelir:
9             fiyat, gelir = p, g
10    return fiyat, gelir

```

DEĞİŞKENLER

v [0.91, 0.78, 0.62, 0.55, 0.47, 0.33, 0.15, 0.08]

$n = 8$

$k = 8$

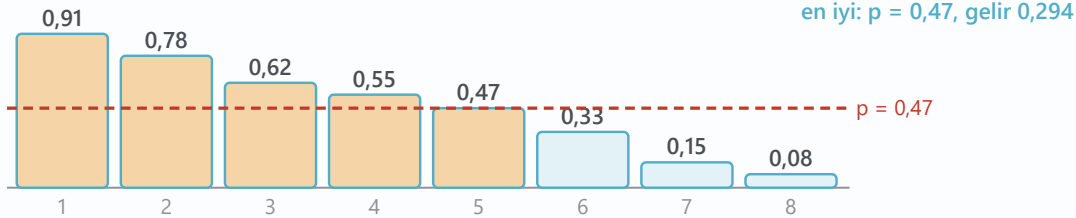
$p = 0.08$

$g = 0.08$

fiyat = 0.47

gelir = 0.294

Kod İzleme: Veriden Rezerv Fiyatı Öğrenmek



Öğrenilen fiyat **0,47**, deneysel gelir 0,294. Değerler aslında $U[0, 1]$ 'den geliyorsa gerçek gelir $0,47 \cdot 0,53 \approx$ **0,249** (en iyisi 0,25).

Adım 12 / 12

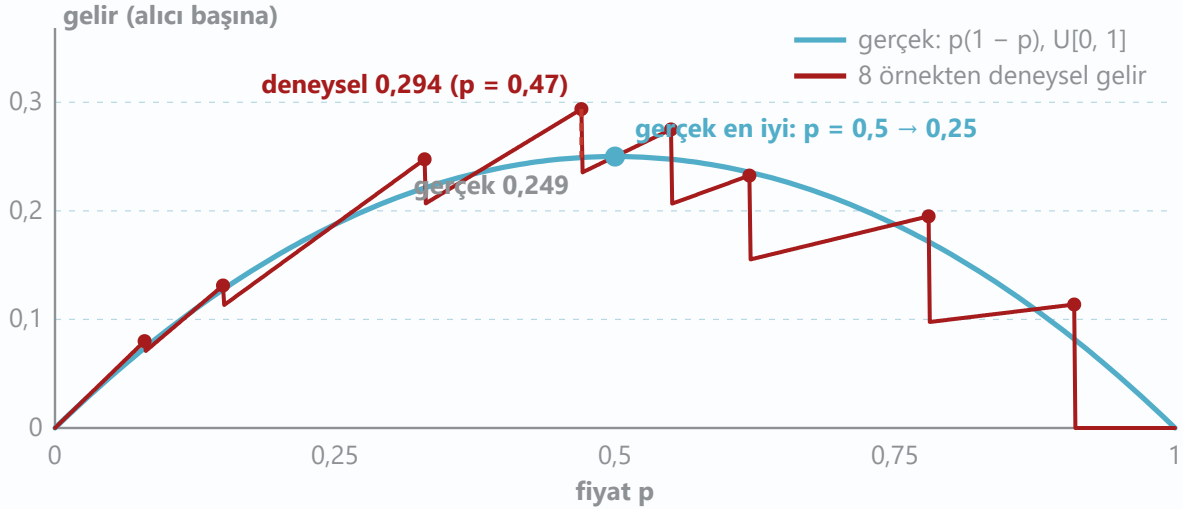
```
1 def en_yii_fiyat(orn):
2     v = sorted(orn, reverse=True)
3     n = len(v)
4     fiyat, gelir = 0, 0
5     for k in range(1, n + 1):
6         p = v[k - 1]          # k. en yüksek
7         g = p * k / n         # k alıcı alır
8         if g > gelir:
9             fiyat, gelir = p, g
10    return fiyat, gelir
```

DEĞİŞKENLER

fiyat = **0.47**

gelir = **0.294**

Deneyisel Gelir Eğrisi Gerçeğinden Ne Kadar Sapar?



- Deneyisel eğri, örnek değerlerde sıçrayan basamaklı bir fonksiyondur; en iyisi her zaman bir örnek değerde çıkar (burada 0,47).
- Seçilen fiyatın örnekteki başarısı (0,294) gerçek başarısını (0,249) abartır: ölçülen ile seçilen aynı veri olduğu için **iyimser yanlılık** oluşur. Gerçek en iyi 0,25 ile fark küçüktür ama örnek azaldıkça büyür.

Kaç Örnek Yeter? Örneklem Karmaşıklığı

örnek n	deneysel gelir	gerçek gelir	0,25 – gerçek
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

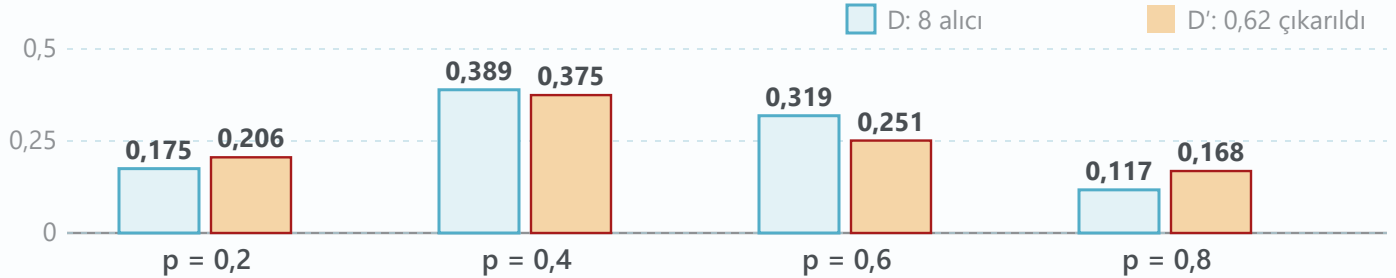
Değerler $U[0, 1]$; her n için yüzlerce tekrarın ortalaması (betikte simülasyon). En iyi gelir 0,25 ($p = 0,5$).

- **Aşırı uyum:** az örnekte deneysel gelir gerçeğinden yüksek görünür; öğrenilen fiyat örneklere “ezberlenmiştir”.
- Örneklem karmaşıklığı: ϵ -yakın gelir için (yüksek olasılıkla) kaç örnek gerekir?
- Düzenli dağılımlarda örnek sayısı $1/\epsilon^2$ 'da polinomdur (Cole ve Roughgarden, 2014); varsayım yoksa (ör. ağır kuyruk) garanti verilemeyebilir.

Ders: veriden öğrenilen mekanizma, az veride kendi başarısını abartır. Eğitim ve değerlendirme verisini ayırmak, makine öğrenmesindeki kadar burada da gereklidir.

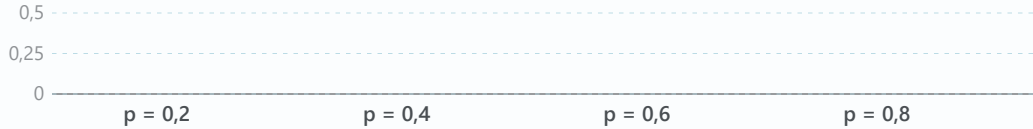
Diferansiyel Mahremiyet: Tek Kişinin Verisi Sonucu Az Etkilesin

- **ϵ -diferansiyel mahremiyet:** tek bir kişinin verisi değişince (komşu veri kümeleri D, D') her çıktının olasılığı en fazla e^ϵ katı değişir.
- **Üstel mekanizma** (McSherry ve Talwar, 2007): aday çıktı r 'yi $\exp(\epsilon \cdot u(D, r) / 2\Delta)$ ile orantılı olasılıkla seç; Δ = bir kişinin u 'yu en fazla değiştirebildiği miktar.
- **Oyun kuramı bağı:** rapor sonucu az etkilediği için yalan söylemenin beklenen kazancı en fazla $(e^\epsilon - 1) \cdot (\text{en büyük fayda})$ olur $\rightarrow$ *yaklaşık* teşvik uyumluluğu, üstelik mahremiyetle birlikte.



Rezerv fiyatı örneği, $\epsilon = 2$: bir alıcı çıkarılınca olasılıklar en fazla 1,44 kat değişir (sınır $e^2 \approx 7,39$). Hesap sonraki slaytta satır satır.

Kod İzleme: Üstel Mekanizma ile Fiyat Seçmek



Veri: 8 alıcının değeri (önceki kod izlemesindeki örnek). Aday rezerv fiyatları 0,2 – 0,8.

Adım 1 / 9

```

1 import math
2 def ustel(v, fiyatlar, eps):
3     R = [p * sum(x >= p for x in v) for p in fiyatlar]
4     w = [math.exp(eps * r / 2) for r in R] # Δ = 1
5     W = sum(w)
6     return [x / W for x in w]
7
8 v = [0.62, 0.15, 0.91, 0.47, 0.33, 0.78, 0.55, 0.08]
9 P = [0.2, 0.4, 0.6, 0.8]
10 q = ustel(v, P, 2)
11 print([round(x, 3) for x in q])
12 q = ustel(v[1:], P, 2) # 0.62 çıkarıldı
13 print([round(x, 3) for x in q])

```

DEĞİŞKENLER

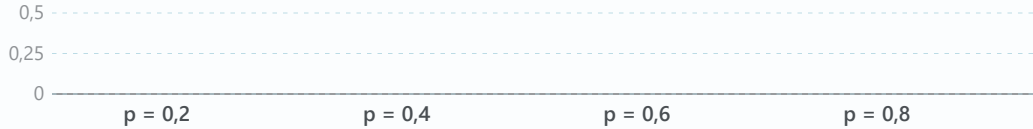
$v = 8$ alıcı

$P = [0.2, 0.4, 0.6, 0.8]$

ÇIKTI

[0.62, 0.15, 0.91, 0.47, 0.33, 0.78, 0.55, 0.08]

Kod İzleme: Üstel Mekanizma ile Fiyat Seçmek



$R = p \cdot (\text{alıcı sayısı})$: $0,2 \cdot 6 = 1,20$, $0,4 \cdot 5 = 2,00$, $0,6 \cdot 3 = 1,80$, $0,8 \cdot 1 = 0,80$.

Adım 2 / 9

```

1 import math
2 def ustel(v, fiyatlar, eps):
3     R = [p * sum(x >= p for x in v) for p in fiyatlar]
4     w = [math.exp(eps * r / 2) for r in R] # Δ = 1
5     W = sum(w)
6     return [x / W for x in w]
7 v = [0.62, 0.15, 0.91, 0.47, 0.33, 0.78, 0.55, 0.08]
8 P = [0.2, 0.4, 0.6, 0.8]
9 q = ustel(v, P, 2)
10 print([round(x, 3) for x in q])
11 q = ustel(v[1:], P, 2) # 0.62 çıkarıldı
12 print([round(x, 3) for x in q])

```

DEĞİŞKENLER

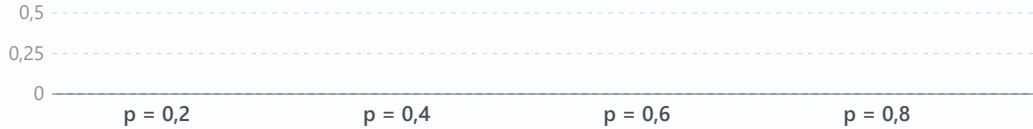
$n = 8$

$R = [1.20, 2.00, 1.80, 0.80]$

ÇIKTI

[0.62, 0.15, 0.91, 0.47, 0.33, 0.78, 0.55, 0.08]

Kod İzleme: Üstel Mekanizma ile Fiyat Seçmek



$w = \exp(\epsilon \cdot R/2) = \exp(R)$: 3,32, 7,39, 6,05, 2,23. Farklar üstel büyür ama kötü fiyatın da şansı kalır.

Adım 3 / 9

```

1 import math
2 def ustel(v, fiyatlar, eps):
3     R = [p * sum(x >= p for x in v) for p in fiyatlar]
4     w = [math.exp(eps * r / 2) for r in R] # Δ = 1
5     W = sum(w)
6     return [x / W for x in w]
7 v = [0.62, 0.15, 0.91, 0.47, 0.33, 0.78, 0.55, 0.08]
8 P = [0.2, 0.4, 0.6, 0.8]
9 q = ustel(v, P, 2)
10 print([round(x, 3) for x in q])
11 q = ustel(v[1:], P, 2) # 0.62 çıkarıldı
12 print([round(x, 3) for x in q])

```

DEĞİŞKENLER

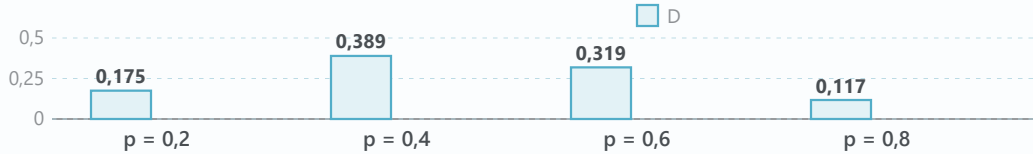
R [1.20, 2.00, 1.80,
= 0.80]

w [3.32, 7.39, 6.05,
= 2.23]

ÇIKTI



Kod İzleme: Üstel Mekanizma ile Fiyat Seçmek



$W = 18,98$; olasılıklar $q = w/W = 0,175, 0,389, 0,319, 0,117$.

Adım 4 / 9

```

1 import math
2 def ustel(v, fiyatlar, eps):
3     R = [p * sum(x >= p for x in v) for p in fiyatlar]
4     w = [math.exp(eps * r / 2) for r in R] # Δ = 1
5     W = sum(w)
6     return [x / W for x in w]
7 v = [0.62, 0.15, 0.91, 0.47, 0.33, 0.78, 0.55, 0.08]
8 P = [0.2, 0.4, 0.6, 0.8]
9 q = ustel(v, P, 2)
10 print([round(x, 3) for x in q])
11 q = ustel(v[1:], P, 2) # 0.62 çıkarıldı
12 print([round(x, 3) for x in q])

```

DEĞİŞKENLER

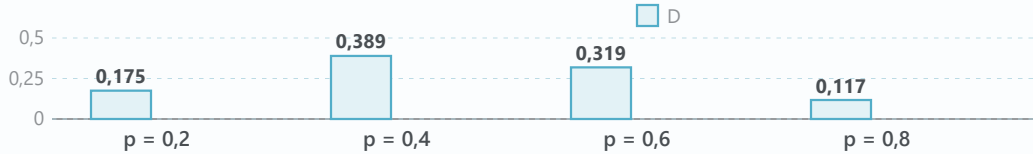
$W = 18.984$

$q = [0.17, 0.39, 0.32, 0.12]$

ÇIKTI

[0.17, 0.39, 0.32, 0.12]

Kod İzleme: Üstel Mekanizma ile Fiyat Seçmek



Çıktı yazdırılır. $p = 0,4$ en olası (%38,9) ama tek başına değil.

Adım 5 / 9

```

1 import math
2 def ustel(v, fiyatlar, eps):
3     R = [p * sum(x >= p for x in v) for p in fiyatlar]
4     w = [math.exp(eps * r / 2) for r in R] # Δ = 1
5     W = sum(w)
6     return [x / W for x in w]
7 v = [0.62, 0.15, 0.91, 0.47, 0.33, 0.78, 0.55, 0.08]
8 P = [0.2, 0.4, 0.6, 0.8]
9 q = ustel(v, P, 2)
10 print([round(x, 3) for x in q])
11 q = ustel(v[1:], P, 2) # 0.62 çıkarıldı
12 print([round(x, 3) for x in q])

```

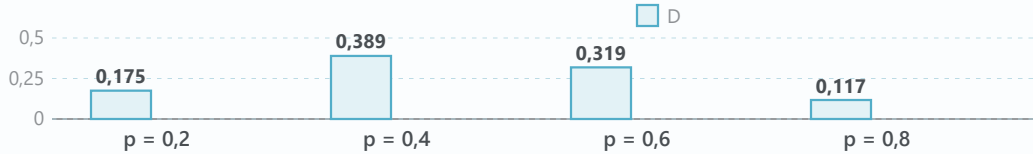
DEĞİŞKENLER

q [0.17, 0.39, 0.32,
= 0.12]

ÇIKTI

[0.175, 0.389, 0.319, 0.117]

Kod İzleme: Üstel Mekanizma ile Fiyat Seçmek



$R = p \cdot (\text{alıcı sayısı})$: $0,2 \cdot 5 = 1,00$, $0,4 \cdot 4 = 1,60$, $0,6 \cdot 2 = 1,20$, $0,8 \cdot 1 = 0,80$. 0,62 gidinece $p = 0,6$ 'nın geliri 0,6 düştü.

Adım 6 / 9

```

1 import math
2 def ustel(v, fiyatlar, eps):
3     R = [p * sum(x >= p for x in v) for p in fiyatlar]
4     w = [math.exp(eps * r / 2) for r in R] # Δ = 1
5     W = sum(w)
6     return [x / W for x in w]
7 v = [0.62, 0.15, 0.91, 0.47, 0.33, 0.78, 0.55, 0.08]
8 P = [0.2, 0.4, 0.6, 0.8]
9 q = ustel(v, P, 2)
10 print([round(x, 3) for x in q])
11 q = ustel(v[1:], P, 2) # 0.62 çıkarıldı
12 print([round(x, 3) for x in q])

```

DEĞİŞKENLER

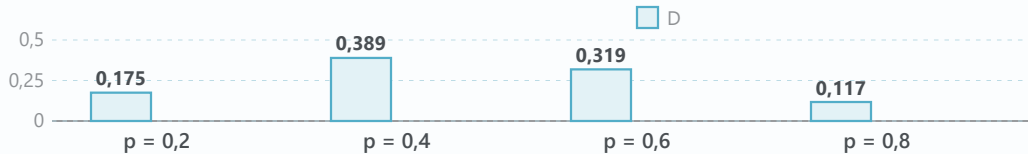
$n = 7$

$R = [1.00, 1.60, 1.20, 0.80]$

ÇIKTI

$[0.175, 0.389, 0.319, 0.117]$

Kod İzleme: Üstel Mekanizma ile Fiyat Seçmek



$w = \exp(\epsilon \cdot R/2) = \exp(R)$: 2,72, 4,95, 3,32, 2,23. Farklar üstel büyür ama kötü fiyatın da şansı kalır.

Adım 7 / 9

```

1 import math
2 def ustel(v, fiyatlar, eps):
3     R = [p * sum(x >= p for x in v) for p in fiyatlar]
4     w = [math.exp(eps * r / 2) for r in R] # Δ = 1
5     W = sum(w)
6     return [x / W for x in w]
7 v = [0.62, 0.15, 0.91, 0.47, 0.33, 0.78, 0.55, 0.08]
8 P = [0.2, 0.4, 0.6, 0.8]
9 q = ustel(v, P, 2)
10 print([round(x, 3) for x in q])
11 q = ustel(v[1:], P, 2) # 0.62 çıkarıldı
12 print([round(x, 3) for x in q])

```

DEĞİŞKENLER

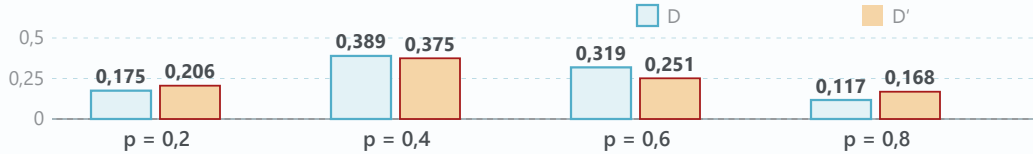
R [1.00, 1.60, 1.20,
= 0.80]

w [2.72, 4.95, 3.32,
= 2.23]

ÇIKTI

[0.175, 0.389, 0.319, 0.117]

Kod İzleme: Üstel Mekanizma ile Fiyat Seçmek



$W = 13,22$; olasılıklar $q = w/W = 0,206, 0,375, 0,251, 0,168$.

Adım 8 / 9

```

1 import math
2 def ustel(v, fiyatlar, eps):
3     R = [p * sum(x >= p for x in v) for p in fiyatlar]
4     w = [math.exp(eps * r / 2) for r in R] # Δ = 1
5     W = sum(w)
6     return [x / W for x in w]
7 v = [0.62, 0.15, 0.91, 0.47, 0.33, 0.78, 0.55, 0.08]
8 P = [0.2, 0.4, 0.6, 0.8]
9 q = ustel(v, P, 2)
10 print([round(x, 3) for x in q])
11 q = ustel(v[1:], P, 2) # 0.62 çıkarıldı
12 print([round(x, 3) for x in q])

```

DEĞİŞKENLER

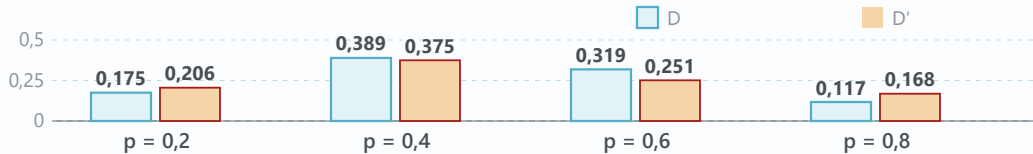
$W = 13.217$

$q = [0.21, 0.37, 0.25, 0.17]$

ÇIKTI

$[0.175, 0.389, 0.319, 0.117]$

Kod İzleme: Üstel Mekanizma ile Fiyat Seçmek



İki dağılım yan yana: en büyük oran $1,44 \leq e^2 = 7,39$. Tek kişinin verisi sonucu sınırlı etkiler: ϵ -diferansiyel mahremiyet.

Adım 9 / 9

```

1 import math
2 def ustel(v, fiyatlar, eps):
3     R = [p * sum(x >= p for x in v) for p in fiyatlar]
4     w = [math.exp(eps * r / 2) for r in R] # Δ = 1
5     W = sum(w)
6     return [x / W for x in w]
7 v = [0.62, 0.15, 0.91, 0.47, 0.33, 0.78, 0.55, 0.08]
8 P = [0.2, 0.4, 0.6, 0.8]
9 q = ustel(v, P, 2)
10 print([round(x, 3) for x in q])
11 q = ustel(v[1:], P, 2) # 0.62 çıkarıldı
▶ print([round(x, 3) for x in q])

```

DEĞİŞKENLER

```

q [0.21, 0.37, 0.25,
= 0.17]

```

ÇIKTI

```

[0.175, 0.389, 0.319, 0.117]
[0.206, 0.375, 0.251, 0.168]

```

Blokzincir ve Akıllı Sözleşmeler

- **Akıllı sözleşme:** kodu herkese açık, zincirde otomatik yürütülen program → kendi kendini uygulayan bir mekanizma.
- Mekanizma tasarımının "tasarımcı kurallara bağlı kalır" (taahhüt) varsayımı kodla sağlanır; ama bekleyen işlemler de herkese görünür.
- Blok üreticisi hangi işlemin bloğa gireceğine ve **sırasına** karar verir: kıt kaynak → açık artırma.
- **İşlem ücreti açık artırması:** Kısıtlı blok alanı teklif verenlere dağıtılır.
- **Merkeziyetsiz borsa:** Otomatik piyasa yapıcı (AMM): fiyat bir formülle belirlenir.



- **Zincir üstü yönetim:** Jeton (token) ağırlıklı oylama; Bölüm 6'nın sorunları yeniden.
- **Doğrulayıcı teşvikleri:** Hisse ispatında (proof of stake) doğrulayıcı teminat yatırır; dürüst davranış ödüllendirilir, kanıtlanabilir hile teminattan kesilir (slashing).

İşlem Ücreti Açık Artırmaları

Birinci fiyat (ilk tasarım)

- Her işlem bir ücret teklif eder; üretici en yüksek teklifleri alır, herkes kendi teklifini öder.
- Değerler 9, 7, 6, 4, 2 ve kapasite 3: dürüst teklifle $9 + 7 + 6 = 22$ öder.
- Ama dürüst teklif en iyi değil: herkes teklifini kırpar (Bölüm 5), "ne kadar vermeliyim?" sorusu kullanıcıyı yorar.

Taban ücret + bahşiş (EIP-1559 türü)

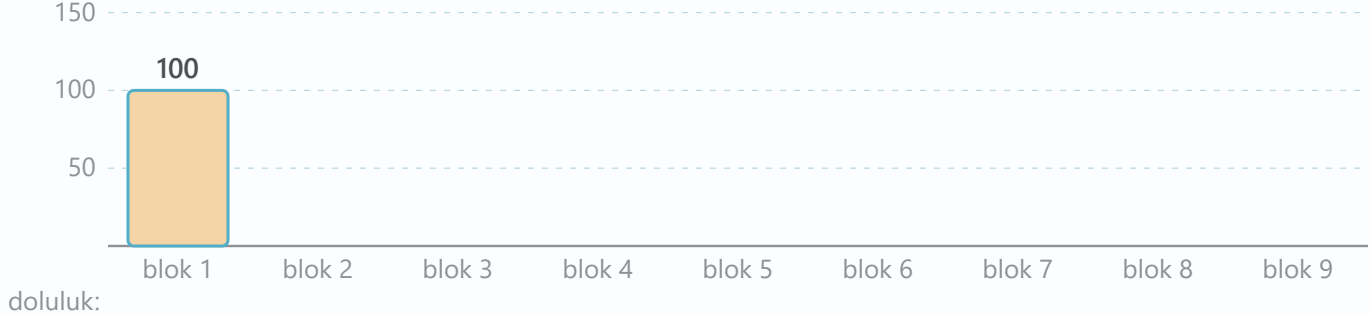
- Protokol bir taban ücret r belirler (burada 5); r **yakılır**, üreticiye yalnız bahşiş (1) gider.
- Değeri $\geq r + \text{bahşiş}$ olanlar girer: 3 işlem, 15 yakılır, 3 bahşiş.
- Blok dolu değilken en iyi strateji basittir: " $r + \text{küçük bahşiş}$ " öde; fiyat tahmini gerekmez.



Tasarım hedefleri (Roughgarden, 2021): kullanıcı için basit teklif, üretici için dürüst davranış, danışıklılığa direnç. Hepsini tam sağlayan kural bulmak zordur.

Adım Adım: Taban Ücret Talebe Göre Ayarlanır

taban ücret r



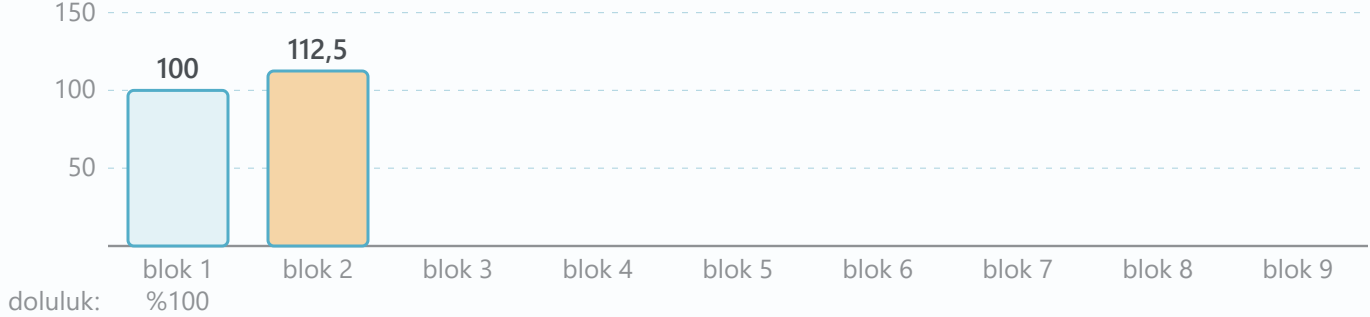
Kural: $r_{t+1} = r_t \cdot (1 + \frac{1}{8} \cdot (g_t - g^*) / g^*)$. Hedef g^* = blok kapasitesinin yarısı. Başlangıç $r = 100$.

Adım 1 / 9

Taban ücret talebi izleyen bir "ilan edilmiş fiyat" gibi çalışır; blok doluyken yine bahşişe dayalı açık artırmaya döner.

Adım Adım: Taban Ücret Talebe Göre Ayarlanır

taban ücret r



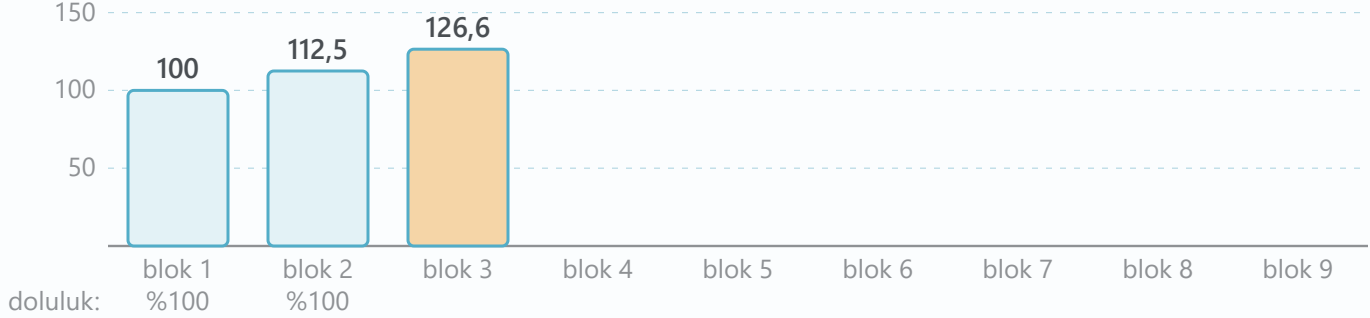
Blok 1 %100 dolu → değişim +%12,5 → r = **112,5**.

Adım 2 / 9

Taban ücret talebi izleyen bir "ilan edilmiş fiyat" gibi çalışır; blok doluyken yine bahşişe dayalı açık artırmaya döner.

Adım Adım: Taban Ücret Talebe Göre Ayarlanır

taban ücret r



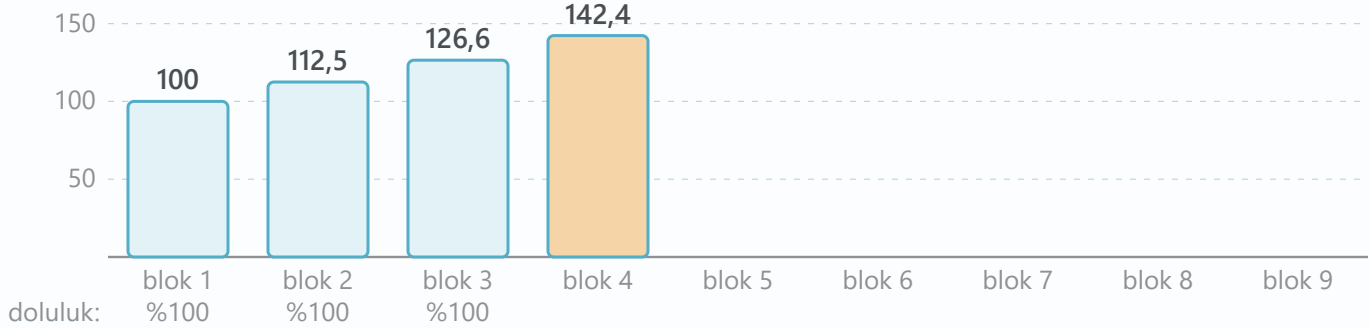
Blok 2 %100 dolu → değişim +%12,5 → r = **126,6**.

Adım 3 / 9

Taban ücret talebi izleyen bir "ilan edilmiş fiyat" gibi çalışır; blok doluyken yine bahşişe dayalı açık artırmaya döner.

Adım Adım: Taban Ücret Talebe Göre Ayarlanır

taban ücret r



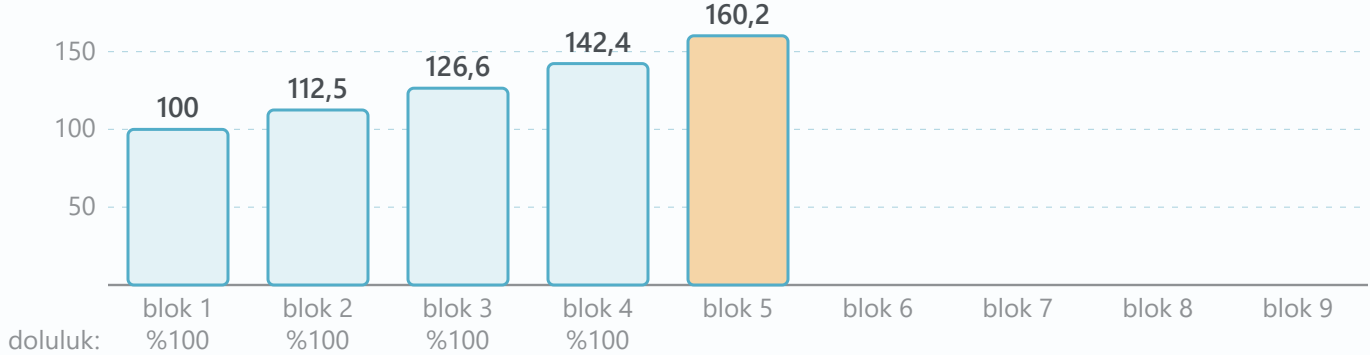
Blok 3 %100 dolu → değişim +%12,5 → r = **142,4**.

Adım 4 / 9

Taban ücret talebi izleyen bir "ilan edilmiş fiyat" gibi çalışır; blok doluyken yine bahşişe dayalı açık artırmaya döner.

Adım Adım: Taban Ücret Talebe Göre Ayarlanır

taban ücret r



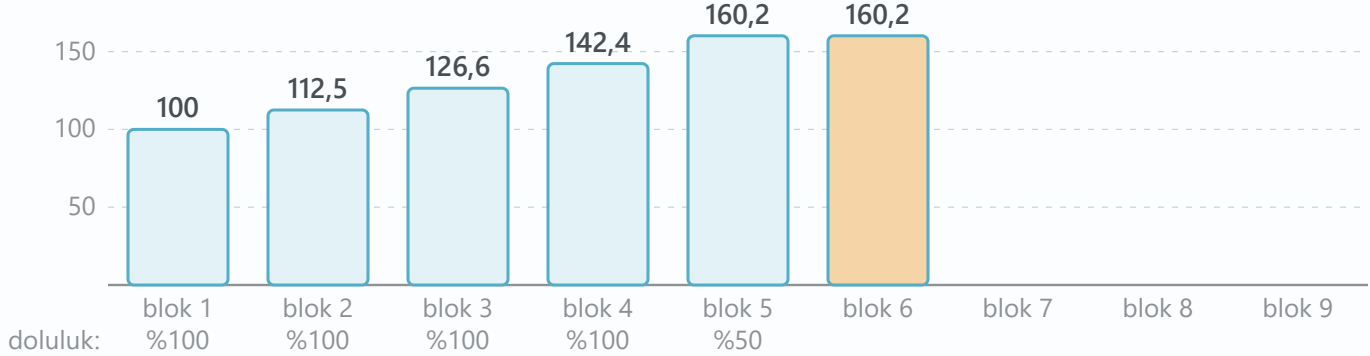
Blok 4 %100 dolu → değişim +%12,5 → r = **160,2**. Talep patlaması: ücret her blokta en çok %12,5 artar.

Adım 5 / 9

Taban ücret talebi izleyen bir "ilan edilmiş fiyat" gibi çalışır; blok doluyken yine bahşişe dayalı açık artırmaya döner.

Adım Adım: Taban Ücret Talebe Göre Ayarlanır

taban ücret r



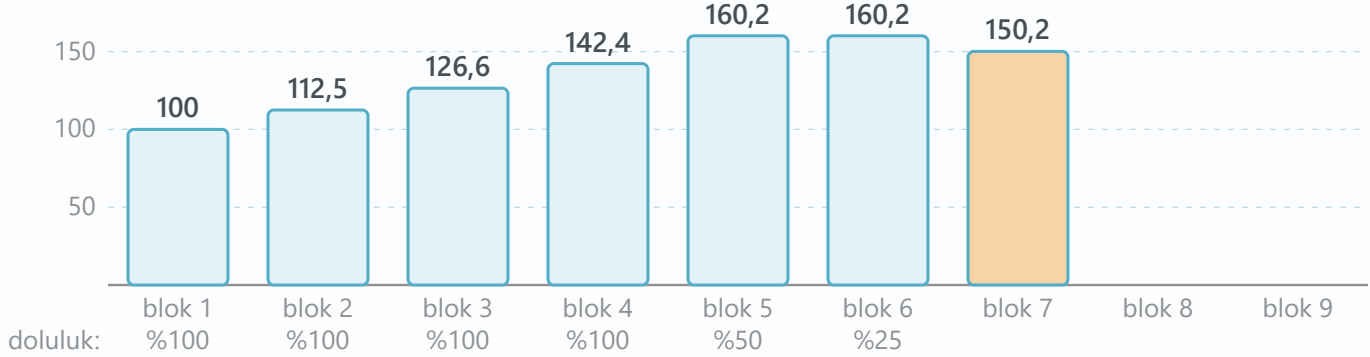
Blok 5 %50 dolu → değişim $\pm\%0$ → r = **160,2**. Tam hedefte ücret sabit kalır.

Adım 6 / 9

Taban ücret talebi izleyen bir "ilan edilmiş fiyat" gibi çalışır; blok doluyken yine bahşişe dayalı açık artırmaya döner.

Adım Adım: Taban Ücret Talebe Göre Ayarlanır

taban ücret r



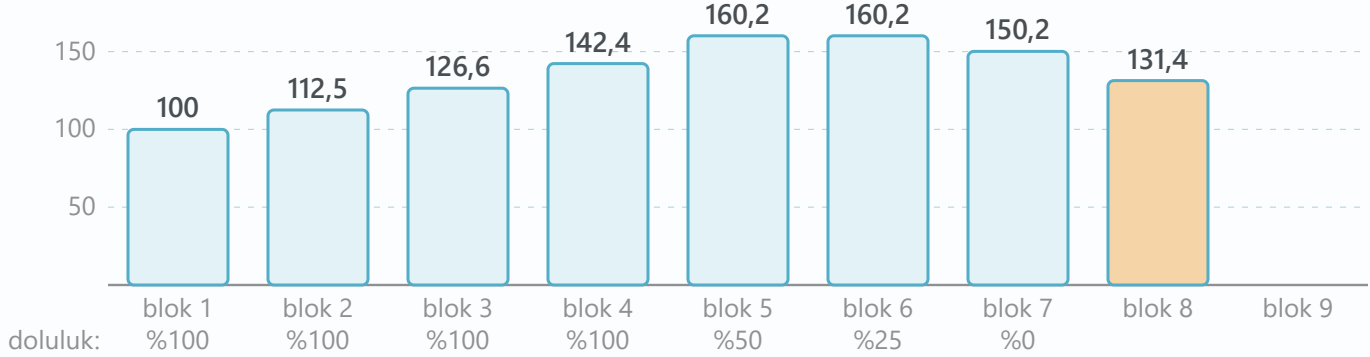
Blok 6 %25 dolu → değişim $-\%6,25 \rightarrow r = 150,2$.

Adım 7 / 9

Taban ücret talebi izleyen bir "ilan edilmiş fiyat" gibi çalışır; blok doluyken yine bahşişe dayalı açık artırmaya döner.

Adım Adım: Taban Ücret Talebe Göre Ayarlanır

taban ücret r



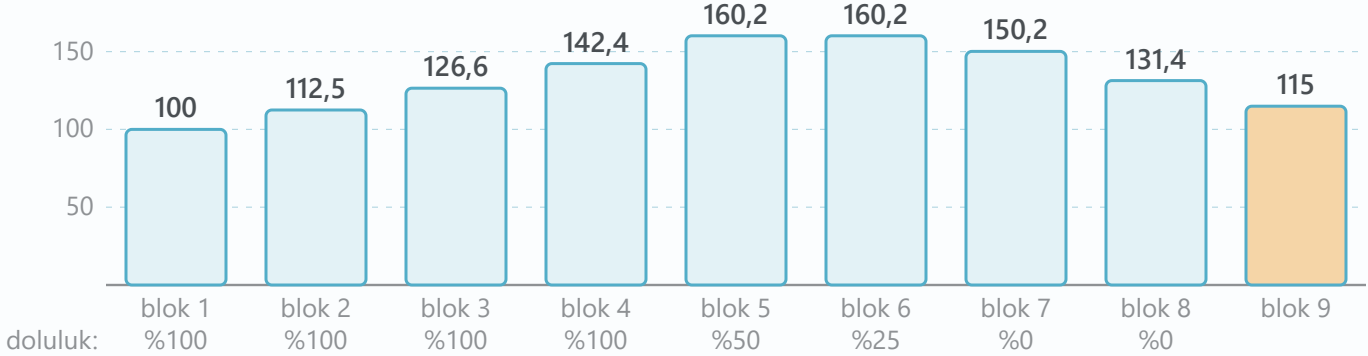
Blok 7 %0 dolu → değişim $-\%12,5$ → r = **131,4**.

Adım 8 / 9

Taban ücret talebi izleyen bir "ilan edilmiş fiyat" gibi çalışır; blok doluyken yine bahşişe dayalı açık artırmaya döner.

Adım Adım: Taban Ücret Talebe Göre Ayarlanır

taban ücret r



Blok 8 %0 dolu → değişim $-\%12,5$ → $r = 115$. Boş bloklar ücreti düşürür; talep azalınca fiyat geri gelir.

Adım 9 / 9

Taban ücret talebi izleyen bir "ilan edilmiş fiyat" gibi çalışır; blok doluyken yine bahşişe dayalı açık artırmaya döner.

Adım Adım: MEV ve Sandviç Saldırısı

Sıra	İşlem	Havuz (A, B)	Sonuç
0	Başlangıç ($x \cdot y = 1\,000\,000$)	(1000; 1000)	

Sabit çarpımlı havuz: A satınca B azalır, fiyat kayar. Kullanıcı 100 A ile tek başına 90,91 B alırdı.

Kullanıcının işlemi bloğa girmeden önce herkese görünür (bekleme havuzu, mempool).

Adım Adım: MEV ve Sandviç Saldırısı

Sıra	İşlem	Havuz (A, B)	Sonuç
0	Başlangıç ($x \cdot y = 1\,000\,000$)	(1000; 1000)	
1	Saldırgan 50 A satar (öne geçme)	(1050; 952,38)	47,62 B alır

Sırayı belirleyebilen biri kendi işlemini kullanıcının **önüne** koyar: B'yi ucuzken alır, fiyatı yukarı iter.

Öne geçme (front-running).

Adım Adım: MEV ve Sandviç Saldırısı

Sıra	İşlem	Havuz (A, B)	Sonuç
0	Başlangıç ($x \cdot y = 1\,000\,000$)	(1000; 1000)	
1	Saldırgan 50 A satar (öne geçme)	(1050; 952,38)	47,62 B alır
2	Kullanıcı 100 A satar	(1150; 869,57)	82,82 B alır (tek başına 90,91)

Kullanıcı artık kötü fiyattan işlem yapar: 82,82 B → **8,09 B kayıp**.

Kullanıcının işlemi fiyatı daha da yukarı iter.

Adım Adım: MEV ve Sandviç Saldırısı

Sıra	İşlem	Havuz (A, B)	Sonuç
0	Başlangıç ($x \cdot y = 1\,000\,000$)	(1000; 1000)	
1	Saldırgan 50 A satar (öne geçme)	(1050; 952,38)	47,62 B alır
2	Kullanıcı 100 A satar	(1150; 869,57)	82,82 B alır (tek başına 90,91)
3	Saldırgan B'leri geri satar (arkadan gelme)	(1090,29; 917,18)	59,71 A alır

Saldırgan B'leri pahalıdan geri satar: 59,71 A → **kâr 9,71 A** (ücretler hariç).

Arkadan gelme (back-running): "sandviç" tamamlandı.

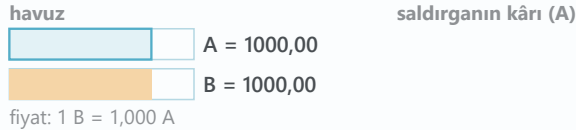
Adım Adım: MEV ve Sandviç Saldırısı

Sıra	İşlem	Havuz (A, B)	Sonuç
0	Başlangıç ($x \cdot y = 1\,000\,000$)	(1000; 1000)	
1	Saldırgan 50 A satar (öne geçme)	(1050; 952,38)	47,62 B alır
2	Kullanıcı 100 A satar	(1150; 869,57)	82,82 B alır (tek başına 90,91)
3	Saldırgan B'leri geri satar (arkadan gelme)	(1090,29; 917,18)	59,71 A alır

MEV (maksimum elde edilebilir değer, maximal extractable value): işlemleri sıralama, ekleme ve çıkarma gücünden elde edilebilecek değer. Önlem fikirleri: toplu açık artırma (aynı fiyat), şifreli bekleme havuzu, sıralama hakkının açık artırılması.

Mekanizma tasarımı sorusu: sıralama gücü kimde, bedelini kim ödüyor?

Kod İzleme: Sandviç Saldırısı ve Kayma Sınırı



Kullanıcı 100 A satacak, en az 82 B istiyor (kayma toleransı). Saldırgan öne geçme miktarı a için üç değer dener.

Adım 1 / 12

```

1 cikti = lambda x, y, dx: y - x * y / (x + dx) # xy=k
2 def sandvic(x, y, a, u, min_out):
3     b1 = cikti(x, y, a) # öne geç: a A sat
4     x, y = x + a, y - b1
5     b2 = cikti(x, y, u) # kullanıcı: u A sat
6     if b2 < min_out:
7         return None # işlem iptal
8     x, y = x + u, y - b2
9     a2 = cikti(y, x, b1) # arkadan: b1 B sat
10    return round(a2 - a, 2)
▶ for a in [20, 50, 80]:
12    print(a, sandvic(1000, 1000, a, 100, 82))

```

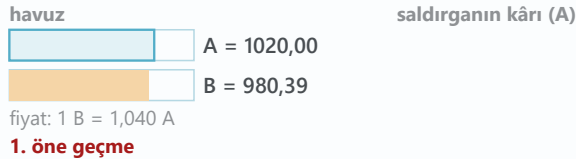
DEĞİŞKENLER

a = 20

ÇIKTI



Kod İzleme: Sandviç Saldırısı ve Kayma Sınırı



a = 20: saldırgan önce 20 A satar, 19,61 B alır. B azaldı → fiyat yükseldi.

Adım 2 / 12

```

1 cikti = lambda x, y, dx: y - x * y / (x + dx) # xy=k
2 def sandvic(x, y, a, u, min_out):
    ▶ b1 = cikti(x, y, a) # öne geç: a A sat
    ▶ x, y = x + a, y - b1
    5 b2 = cikti(x, y, u) # kullanıcı: u A sat
    6 if b2 < min_out:
    7     return None # işlem iptal
    8 x, y = x + u, y - b2
    9 a2 = cikti(y, x, b1) # arkadan: b1 B sat
    10 return round(a2 - a, 2)
11 for a in [20, 50, 80]:
12     print(a, sandvic(1000, 1000, a, 100, 82))

```

DEĞİŞKENLER

a = 20

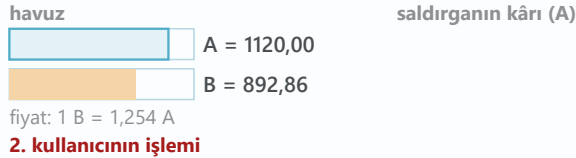
b1 = 19.61

x = 1020.0

y = 980.39

ÇIKTI

Kod İzleme: Sandviç Saldırısı ve Kayma Sınırı



Kullanıcı 87,54 B alır (≥ 82 , işlem geçer; tek başına 90,91 alırdı).

Adım 3 / 12

```

1 cikti = lambda x, y, dx: y - x * y / (x + dx) # xy=k
2 def sandvic(x, y, a, u, min_out):
3     b1 = cikti(x, y, a) # öne geç: a A sat
4     x, y = x + a, y - b1
5     ▶ b2 = cikti(x, y, u) # kullanıcı: u A sat
6     ▶ if b2 < min_out:
7         return None # işlem iptal
8     ▶ x, y = x + u, y - b2
9     a2 = cikti(y, x, b1) # arkadan: b1 B sat
10    return round(a2 - a, 2)
11 for a in [20, 50, 80]:
12    print(a, sandvic(1000, 1000, a, 100, 82))

```

DEĞİŞKENLER

a = 20

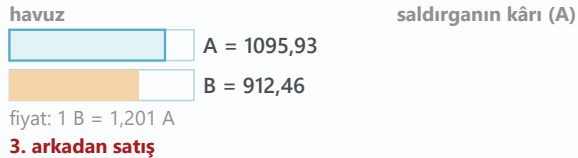
b2 = 87.54

x = 1120.0

y = 892.86

ÇIKTI

Kod İzleme: Sandviç Saldırısı ve Kayma Sınırı



Saldırgan 19,61 B'yi geri satar, 24,07 A alır: $kâr = 24,07 - 20 = \mathbf{4,07 A}$.

Adım 4 / 12

```

1 cikti = lambda x, y, dx: y - x * y / (x + dx) # xy=k
2 def sandvic(x, y, a, u, min_out):
3     b1 = cikti(x, y, a) # öne geç: a A sat
4     x, y = x + a, y - b1
5     b2 = cikti(x, y, u) # kullanıcı: u A sat
6     if b2 < min_out:
7         return None # işlem iptal
8     x, y = x + u, y - b2
9     a2 = cikti(y, x, b1) # arkadan: b1 B sat
10    return round(a2 - a, 2)
11 for a in [20, 50, 80]:
12     print(a, sandvic(1000, 1000, a, 100, 82))

```

DEĞİŞKENLER

a = 20

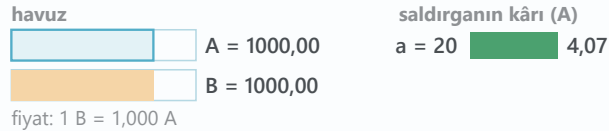
a2 = 24.07

kar = 4.07

ÇIKTI



Kod İzleme: Sandviç Saldırısı ve Kayma Sınırı



Sonuç yazdırılır.

Adım 5 / 12

```

1 cikti = lambda x, y, dx: y - x * y / (x + dx) # xy=k
2 def sandvic(x, y, a, u, min_out):
3     b1 = cikti(x, y, a) # öne geç: a A sat
4     x, y = x + a, y - b1
5     b2 = cikti(x, y, u) # kullanıcı: u A sat
6     if b2 < min_out:
7         return None # işlem iptal
8     x, y = x + u, y - b2
9     a2 = cikti(y, x, b1) # arkadan: b1 B sat
10    return round(a2 - a, 2)

▶ for a in [20, 50, 80]:
▶     print(a, sandvic(1000, 1000, a, 100, 82))

```

DEĞİŞKENLER

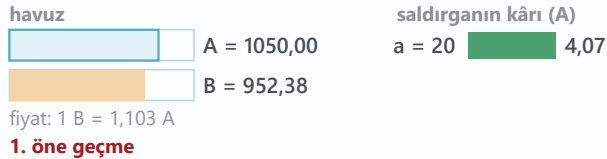
a = 20

sonuc = 4.07

ÇIKTI

20 4.07

Kod İzleme: Sandviç Saldırısı ve Kayma Sınırı



a = 50: saldırgan önce 50 A satar, 47,62 B alır. B azaldı → fiyat yükseldi.

Adım 6 / 12

```

1 cikti = lambda x, y, dx: y - x * y / (x + dx) # xy=k
2 def sandvic(x, y, a, u, min_out):
    ▶ b1 = cikti(x, y, a) # öne geç: a A sat
    ▶ x, y = x + a, y - b1
    5 b2 = cikti(x, y, u) # kullanıcı: u A sat
    6 if b2 < min_out:
    7     return None # işlem iptal
    8 x, y = x + u, y - b2
    9 a2 = cikti(y, x, b1) # arkadan: b1 B sat
    10 return round(a2 - a, 2)
    11 for a in [20, 50, 80]:
    12     print(a, sandvic(1000, 1000, a, 100, 82))

```

DEĞİŞKENLER

a = 50

b1 = 47.62

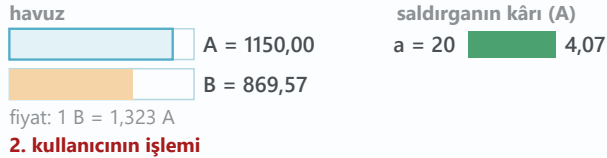
x = 1050.0

y = 952.38

ÇIKTI

20 4.07

Kod İzleme: Sandviç Saldırısı ve Kayma Sınırı



Kullanıcı 82,82 B alır (≥ 82 , işlem geçer; tek başına 90,91 alırdı).

Adım 7 / 12

```
1 cikti = lambda x, y, dx: y - x * y / (x + dx) # xy=k
2 def sandvic(x, y, a, u, min_out):
3     b1 = cikti(x, y, a) # öne geç: a A sat
4     x, y = x + a, y - b1
5     ▶ b2 = cikti(x, y, u) # kullanıcı: u A sat
6     ▶ if b2 < min_out:
7         return None # işlem iptal
8     ▶ x, y = x + u, y - b2
9     a2 = cikti(y, x, b1) # arkadan: b1 B sat
10    return round(a2 - a, 2)
11 for a in [20, 50, 80]:
12    print(a, sandvic(1000, 1000, a, 100, 82))
```

DEĞİŞKENLER

a = 50

b2 = 82.82

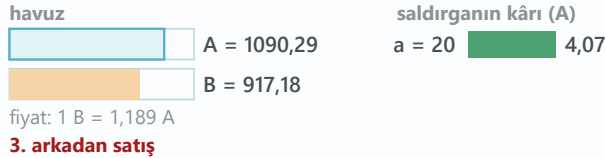
x = 1150.0

y = 869.57

ÇIKTI

20 4.07

Kod İzleme: Sandviç Saldırısı ve Kayma Sınırı



Saldırgan 47,62 B'yi geri satar, 59,71 A alır: kâr = 59,71 – 50 = **9,71 A**.

Adım 8 / 12

```

1 cikti = lambda x, y, dx: y - x * y / (x + dx) # xy=k
2 def sandvic(x, y, a, u, min_out):
3     b1 = cikti(x, y, a) # öne geç: a A sat
4     x, y = x + a, y - b1
5     b2 = cikti(x, y, u) # kullanıcı: u A sat
6     if b2 < min_out:
7         return None # işlem iptal
8     x, y = x + u, y - b2
9     a2 = cikti(y, x, b1) # arkadan: b1 B sat
10    return round(a2 - a, 2)
11 for a in [20, 50, 80]:
12     print(a, sandvic(1000, 1000, a, 100, 82))

```

DEĞİŞKENLER

a = 50

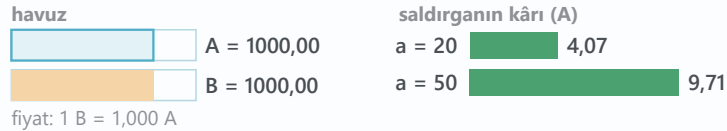
a2 = 59.71

kar = 9.71

ÇIKTI

20 4.07

Kod İzleme: Sandviç Saldırısı ve Kayma Sınırı



Sonuç yazdırılır.

Adım 9 / 12

```

1 cikti = lambda x, y, dx: y - x * y / (x + dx) # xy=k
2 def sandvic(x, y, a, u, min_out):
3     b1 = cikti(x, y, a)           # öne geç: a A sat
4     x, y = x + a, y - b1
5     b2 = cikti(x, y, u)           # kullanıcı: u A sat
6     if b2 < min_out:
7         return None               # işlem iptal
8     x, y = x + u, y - b2
9     a2 = cikti(y, x, b1)          # arkadan: b1 B sat
10    return round(a2 - a, 2)

▶ for a in [20, 50, 80]:
▶     print(a, sandvic(1000, 1000, a, 100, 82))

```

DEĞİŞKENLER

a = 50

sonuc = 9.71

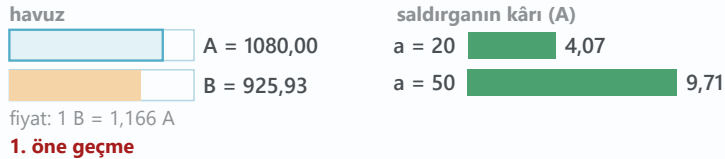
ÇIKTI

```

20 4.07
50 9.71

```

Kod İzleme: Sandviç Saldırısı ve Kayma Sınırı



a = 80: saldırgan önce 80 A satar, 74,07 B alır. B azaldı → fiyat yükseldi.

Adım 10 / 12

```

1 cikti = lambda x, y, dx: y - x * y / (x + dx) # xy=k
2 def sandvic(x, y, a, u, min_out):
    ▶ b1 = cikti(x, y, a) # öne geç: a A sat
    ▶ x, y = x + a, y - b1
    5 b2 = cikti(x, y, u) # kullanıcı: u A sat
    6 if b2 < min_out:
    7     return None # işlem iptal
    8 x, y = x + u, y - b2
    9 a2 = cikti(y, x, b1) # arkadan: b1 B sat
    10 return round(a2 - a, 2)
    11 for a in [20, 50, 80]:
    12     print(a, sandvic(1000, 1000, a, 100, 82))

```

DEĞİŞKENLER

a = 80

b1 = 74.07

x = 1080.0

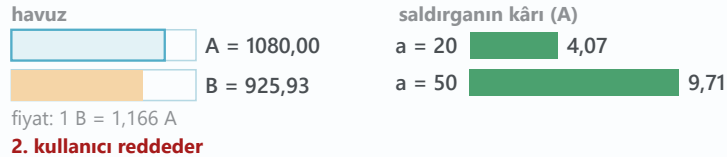
y = 925.93

ÇIKTI

20 4.07

50 9.71

Kod İzleme: Sandviç Saldırısı ve Kayma Sınırı



Kullanıcı yalnızca 78,47 B alırdı < 82 → işlem **iptal**, saldırı başarısız (None).

Adım 11 / 12

```

1 cikti = lambda x, y, dx: y - x * y / (x + dx) # xy=k
2 def sandvic(x, y, a, u, min_out):
3     b1 = cikti(x, y, a) # öne geç: a A sat
4     x, y = x + a, y - b1
5     b2 = cikti(x, y, u) # kullanıcı: u A sat
6     if b2 < min_out:
7         return None # işlem iptal
8     x, y = x + u, y - b2
9     a2 = cikti(y, x, b1) # arkadan: b1 B sat
10    return round(a2 - a, 2)
11 for a in [20, 50, 80]:
12    print(a, sandvic(1000, 1000, a, 100, 82))
  
```

DEĞİŞKENLER

a = 80

b2 = 78.47

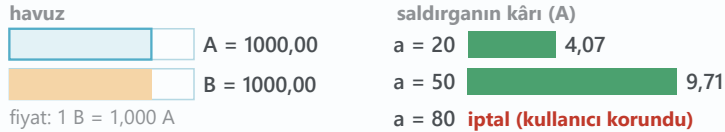
min_out = 82

ÇIKTI

20 4.07

50 9.71

Kod İzleme: Sandviç Saldırısı ve Kayma Sınırı



Daha büyük öne geçme daha çok kâr getirir, ama kullanıcının kayma sınırı onu keser: saldırganın en iyisi sınırı tam zorlayan a. Sıkı tolerans MEV'e karşı basit bir korumadır.

Adım 12 / 12

```
1 cikti = lambda x, y, dx: y - x * y / (x + dx) # xy=k
2 def sandvic(x, y, a, u, min_out):
3     b1 = cikti(x, y, a) # öne geç: a A sat
4     x, y = x + a, y - b1
5     b2 = cikti(x, y, u) # kullanıcı: u A sat
6     if b2 < min_out:
7         return None # işlem iptal
8     x, y = x + u, y - b2
9     a2 = cikti(y, x, b1) # arkadan: b1 B sat
10    return round(a2 - a, 2)
▶ for a in [20, 50, 80]:
▶     print(a, sandvic(1000, 1000, a, 100, 82))
```

DEĞİŞKENLER

a = 80

sonuc = None

ÇIKTI

20 4.07
50 9.71
80 None

Hesaplama ve Ölçek

- Büyük oyunlarda denge bulmak zor olabilir (PPAD, Bölüm 3) → **yaklaşık denge** ve öğrenme dinamikleri (Bölüm 12).
- Çok oyunculu pokerde insan üstü sonuçlar pişmanlığı en aza indirme ve arama ile elde edildi (Brown ve Sandholm, 2019).
- Kombinatoryal açık artırmada kazananı belirlemek NP-zordur; teklif dili ve yaklaşık algoritmalar gerekir.
- Kuramsal garanti kadar gecikme, veri erişimi ve enerji maliyeti de önemlidir.

nesne m	paket sayısı $2^m - 1$
5	31
10	1.023
20	1.048.575
30	1.073.741.823

Her teklifçi her pakete değer bildirirse girdi üstel büyür. İki eylemli n oyunculu oyunda strateji profili 2^n : n = 20 için 1.048.576.

Diğer Araştırma Yönleri

- **Sosyal ağlarda etki en büyükleme:** bir firma k kişiye ücretsiz ürün verecek; yayılmayı en büyükleyecek k "tohum" kişi kimdir? (Kempe, Kleinberg ve Tardos, 2003)
- **Bağımsız çağlayan modeli:** yeni etkilenen her kişi, her komşusunu yalnızca bir kez, p olasılıkla etkiler.
- En iyi tohum kümesini bulmak NP-zordur; ama beklenen yayılım **azalan marjinal getirili** (alt modüler) olduğu için her adımda en çok katkı yapanı ekleyen açgözlü seçim, en iyinin yaklaşık $1 - 1/e \approx \%63$ 'ünü garanti eder.

Çok ajanlı pekiştirmeli öğrenme

Her ajanın ortamı öğrenen diğer ajanları da içerir; ortam durağan olmadığı için tek ajanlı yakınsama garantileri geçerli değildir (Bölüm 12).

Sürdürülebilir sistemler

Karbon emisyon izni piyasaları, elektrikte talep yanıtı: dışsallığı fiyatlayan mekanizmalar (Bölüm 13'teki Pigou vergisi).

Dayanıklılık

Saldırgan, hatalı ya da sınırlı rasyonel oyunculara rağmen iyi çalışan mekanizmalar.

Açık Araştırma Soruları

İnsan + yapay zekâ pazarları

İnsanlar ve ajanlar aynı pazarda etkileştiğinde kurallar nasıl değişmeli?

Ödünleşimleri ölçmek

Verimlilik, mahremiyet ve adalet arasındaki ödünleşimler nasıl ölçülmeli?

Uyarlanır dayanıklılık

Manipülasyona dayanıklı mekanizmalar değişen davranışlara nasıl uyarlanır?

Kuram ve veri

Modellerin varsayımları gerçek platform verileriyle nasıl sınanır?

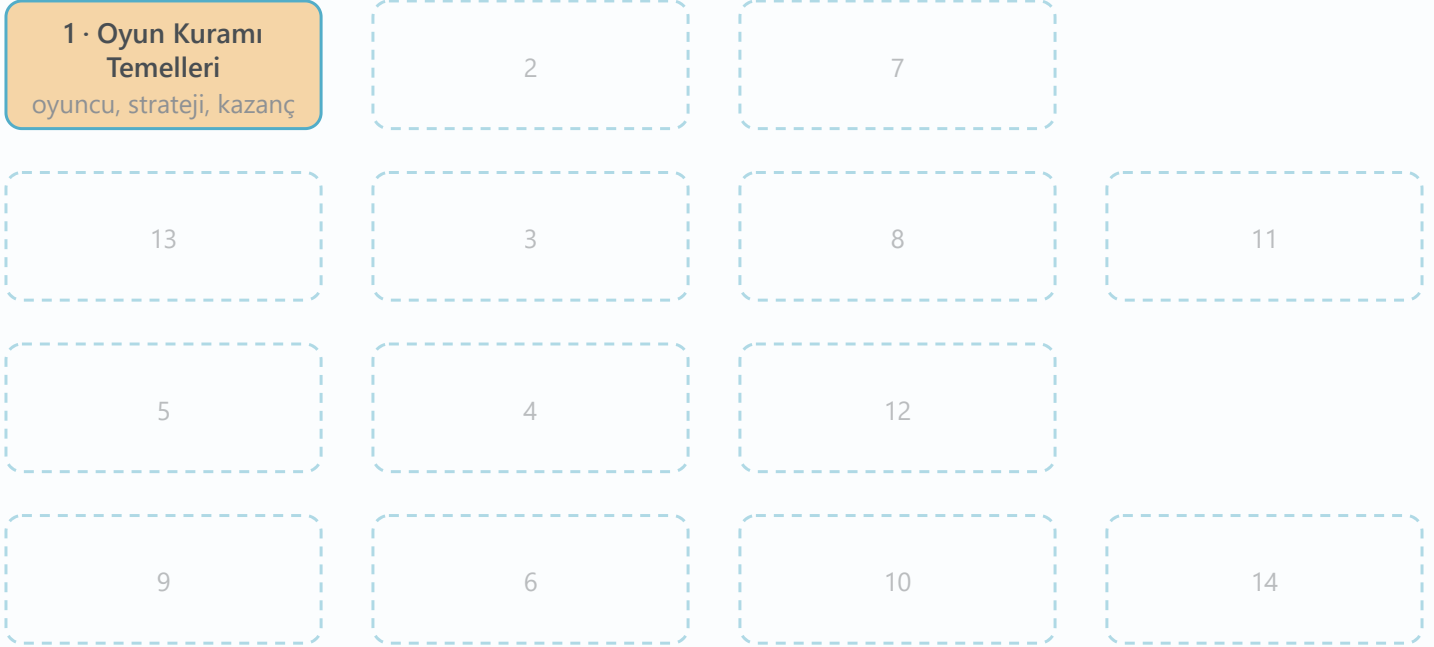
Öğrenen ajanların dengesi

Birbirine karşı öğrenen algoritmalar hangi sonuca yakınsar; danışıklılık nasıl önlenir?

Doğrulanabilir mekanizma

Öğrenilmiş bir kuralın teşvik uyumlu olduğu nasıl kanıtlanır veya denetlenir?

Dersin Haritası: 14 Bölüm, Tek Resim



Bölüm 1 · Oyun Kuramı Temelleri: oyuncu, strateji, kazanç, rasyonellik; Mahkûmun İkilemi. Her şey buradan başlar.

Dersin Haritası: 14 Bölüm, Tek Resim



Bölüm 2 · Oyun Kuramına Giriş ← Bölüm 1: temel kavramlar (oyuncu, strateji, kazanç) oyun türlerine ve temsillerine ayrılır.

Dersin Haritası: 14 Bölüm, Tek Resim



Bölüm 3 · Nash Dengesi ← Bölüm 2: 2×2 oyunlarda işaretlenen dengeler genel Nash tanımına ve varlık teoremine götürür.

Dersin Haritası: 14 Bölüm, Tek Resim



Bölüm 4 · Mekanizma Tasarımı ← Bölüm 3: Denge + teşvik: oyunu çözmek yerine istenen dengeyi veren kurallı tasarlamak.

Dersin Haritası: 14 Bölüm, Tek Resim



Bölüm 5 · Açık Artırmalar ← Bölüm 4: VCG'nin tek nesneli hâli ikinci fiyat (Vickrey) açık artırmasıdır.

Dersin Haritası: 14 Bölüm, Tek Resim



Bölüm 6 · Oylama ← Bölüm 4: Sosyal seçim fonksiyonu → oylama kuralları; Gibbard-Satterthwaite.

Dersin Haritası: 14 Bölüm, Tek Resim



Bölüm 7 · Çoklu Ajan Sistemleri ← Bölüm 2: İşbirlikçi / işbirlikçi olmayan ayrımı → koalisyon oyunları, Shapley değeri.

Dersin Haritası: 14 Bölüm, Tek Resim



Bölüm 8 · Ağ Tasarımı ve Yönlendirme ← Bölüm 3: Tıkanıklık oyunlarında Nash dengesi = Wardrop dengesi; Rosenthal potansiyeli.

Dersin Haritası: 14 Bölüm, Tek Resim



Dersin Haritası: 14 Bölüm, Tek Resim



Bölüm 10 · Algoritmik Adalet ve Etik ← Bölüm 6: Kolektif karardan adalet: ölçütler ve adil bölüşüm.

Dersin Haritası: 14 Bölüm, Tek Resim



Bölüm 11 · Anarşinin Bedeli ← Bölüm 8: Pigou ve Braess ağlarında denge kalitesi → Anarşinin Bedeli 4/3.

Dersin Haritası: 14 Bölüm, Tek Resim



Bölüm 12 · Öğrenme ve Adaptasyon ← Bölüm 3: Denge nasıl bulunur? Oyuncular öğrenerek dengeye yakınsayabilir.

Dersin Haritası: 14 Bölüm, Tek Resim



Bölüm 13 · Karar Alma ← Bölüm 1: Rasyonellik varsayımının tek kişilik hâli: beklenen fayda, Bayes güncellemesi.

Dersin Haritası: 14 Bölüm, Tek Resim



Bölüm 14 · Gelecek Yönelimler: ölçek (11), öğrenen ajanlar (12) ve etik (10) bugünün açık sorularında birleşir.

Proje Yönergesi: Beş Aşama

1. Konu seç: bir oyun, mekanizma ya da platform; derste gördüğümüz bir bölüme bağla (haritadan).

Adım 1 / 5

Araçlar

Python ve Jupyter (hızlı deneme, grafik), JavaScript (tarayıcıda simülatör); dengeyi [Nashpy](#) veya [Gambit](#) ile doğrulayın.

Zaman ipuçları

Erken başlayın; önce basit bir konsol sürümü yapın, özellikleri sonra ekleyin; kodu sık test edin.

Ek puan

Kullanıcının kazançları girip dengeyi görsel olarak izlediği bir simülatör ya da kısa bir makalenin özeti ve basit kodlaması.

Proje Yönergesi: Beş Aşama

- 1. Konu seç:** bir oyun, mekanizma ya da platform; derste gördüğümüz bir bölüme bağla (haritadan).
- 2. Modelle:** oyuncular, stratejiler, kazanç matrisi veya fonksiyonu, bilgi yapısı.

Adım 2 / 5

Araçlar

Python ve Jupyter (hızlı deneme, grafik), JavaScript (tarayıcıda simülatör); dengeyi [Nashpy](#) veya [Gambit](#) ile doğrulayın.

Zaman ipuçları

Erken başlayın; önce basit bir konsol sürümü yapın, özellikleri sonra ekleyin; kodu sık test edin.

Ek puan

Kullanıcının kazançları girip dengeyi görsel olarak izlediği bir simülatör ya da kısa bir makalenin özeti ve basit kodlaması.

Proje Yönergesi: Beş Aşama

- 1. Konu seç:** bir oyun, mekanizma ya da platform; derste gördüğümüz bir bölüme bağla (haritadan).
- 2. Modelle:** oyuncular, stratejiler, kazanç matrisi veya fonksiyonu, bilgi yapısı.
- 3. Analiz et:** Nash dengesi, Pareto optimallik, teşvik uyumluluğu, PoA... Sayıları elle ve kodla doğrula.

Adım 3 / 5

Araçlar

Python ve Jupyter (hızlı deneme, grafik), JavaScript (tarayıcıda simülatör); dengeyi [Nashpy](#) veya [Gambit](#) ile doğrulayın.

Zaman ipuçları

Erken başlayın; önce basit bir konsol sürümü yapın, özellikleri sonra ekleyin; kodu sık test edin.

Ek puan

Kullanıcının kazançları girip dengeyi görsel olarak izlediği bir simülatör ya da kısa bir makalenin özeti ve basit kodlaması.

Proje Yönergesi: Beş Aşama

- 1. Konu seç:** bir oyun, mekanizma ya da platform; derste gördüğümüz bir bölüme bağla (haritadan).
- 2. Modelle:** oyuncular, stratejiler, kazanç matrisi veya fonksiyonu, bilgi yapısı.
- 3. Analiz et:** Nash dengesi, Pareto optimallik, teşvik uyumluluğu, PoA... Sayıları elle ve kodla doğrula.
- 4. Uygula:** etkileşimli bir web uygulaması (ör. JavaScript, Python); kullanıcı oynasın, sonucu görsün.

Adım 4 / 5

Araçlar

Python ve Jupyter (hızlı deneme, grafik), JavaScript (tarayıcıda simülatör); dengeyi [Nashpy](#) veya [Gambit](#) ile doğrulayın.

Zaman ipuçları

Erken başlayın; önce basit bir konsol sürümü yapın, özellikleri sonra ekleyin; kodu sık test edin.

Ek puan

Kullanıcının kazançları girip dengeyi görsel olarak izlediği bir simülatör ya da kısa bir makalenin özeti ve basit kodlaması.

Proje Yönergesi: Beş Aşama

- 1. Konu seç:** bir oyun, mekanizma ya da platform; derste gördüğümüz bir bölüme bağla (haritadan).
- 2. Modelle:** oyuncular, stratejiler, kazanç matrisi veya fonksiyonu, bilgi yapısı.
- 3. Analiz et:** Nash dengesi, Pareto optimallik, teşvik uyumluluğu, PoA... Sayıları elle ve kodla doğrula.
- 4. Uygula:** etkileşimli bir web uygulaması (ör. JavaScript, Python); kullanıcı oynasın, sonucu görsün.
- 5. Anlat:** blog yazısı veya rapor; model, algoritma, gerçek dünya bağlantısı. Kaynak kodu herkese açık depoda.

Adım 5 / 5

Araçlar

Python ve Jupyter (hızlı deneme, grafik), JavaScript (tarayıcıda simülatör); dengeyi [Nashpy](#) veya [Gambit](#) ile doğrulayın.

Zaman ipuçları

Erken başlayın; önce basit bir konsol sürümü yapın, özellikleri sonra ekleyin; kodu sık test edin.

Ek puan

Kullanıcının kazançları girip dengeyi görsel olarak izlediği bir simülatör ya da kısa bir makalenin özeti ve basit kodlaması.

Proje Konusu Önerileri

LLM ajan turnuvası (Bölüm 12, 14)

Dil modeli ajanlarını tekrarlanan Mahkûmun İkilemi'nde Tit-for-Tat'a karşı yarıştıran.

Reklam açık artırması (Bölüm 9)

GSP ve VCG'yi aynı tekliflerle çalıştıran, en iyi tepkiyi arayan bir simülatör.

Tıkanıklık görselleştirici (Bölüm 8, 11)

Braess ağına yol ekleyip dengeyi ve PoA'yı canlı hesaplayın.

Kararlı eşleştirme (Bölüm 4)

Gale–Shapley'i öğrenci–okul verisiyle adım adım gösterin.

Oylama karşılaştırmacı (Bölüm 6)

Aynı tercih profilinde beş kuralı sayın, Condorcet döngüsü arayın.

Taban ücret piyasası (Bölüm 14)

Talep şoklarında EIP-1559 türü taban ücretin davranışını simüle edin.

İyi bir proje: doğru model, test edilmiş hesap (denge ve ödemeler betikle doğrulanmış), anlaşılır arayüz, açık anlatım.

Özet

Konu	Ana fikir	Bu bölümdeki sayı
Ajanlar arası pazarlık	Geriye doğru çıkarım; sabır ve son söz güçtür	$T = 4$: A 18,1; sonsuz ufuk 52,6
Algoritmik danışıklılık	Tetik stratejisi δ yeterince büyükse işbirliğini sürdürür	$\delta \geq 1/2$
Veriden fiyat	Deneyssel gelir en büyükleme; az örnekte aşırı uyum	$p = 0,47$: deneyssel 0,294, gerçek 0,249
Üstel mekanizma	Olasılık $\propto \exp(\varepsilon \cdot \text{gelir} / 2\Delta)$; mahremiyet + yaklaşık IC	$\varepsilon = 2$: oran $\leq e^2 \approx 7,39$ (örnekte 1,44)
Taban ücret	Dolu blokta +%12,5, boş blokta -%12,5; ücret yakılır	$100 \rightarrow 160,2 \rightarrow 115$
MEV	Sıralama gücü değer taşır; sandviç saldırısı	kullanıcı -8,09 B, saldırgan +9,71 A
Harita	Denge $\rightarrow$ tasarım $\rightarrow$ uygulama $\rightarrow$ öğrenme $\rightarrow$ gelecek	14 bölüm, 15 kavram bağı

Örnek Problemler

1. 100 TL, iki tur (A sonra B önerir), $\delta = 0,9$. Alt oyun mükemmel dengede paylaşım nedir? Sonsuz ufukta A'nın payı?
2. Üç firma tekel kârını (100) eşit paylaşıyor; sapan bir dönem 100 alıyor, sonra 0. Tetik stratejisi hangi δ için sürer?
3. Örnek değerler 0,9; 0,6; 0,5; 0,2. Deneysel geliri en büyükleen fiyat ve gelir nedir?
4. $r = 100$ iken üç dolu blok ve bir boş blok gelirse taban ücret kaç olur?
5. Haritada Bölüm 9'daki GSP–VCG karşılaştırması hangi iki bölümün kavramlarını birleştirir?
6. Üstel mekanizmada iki aday fiyatın gelirleri 2 ve 1, $\varepsilon = 2$, $\Delta = 1$. Olasılıklar nedir?

► Cevapları göster

Etkileşimli Oyunlar ve Projeler

Yapay Zekâ Ajan Turnuvası

Sabit, uyarlanır ve öğrenen ajanları Mahkûmun İkilemi, Yazı-Tura Eşleştirme ve TKM'de yarıştırın.

Age of Çotanak

Öğrenci projesinden doğan bölge kontrol oyunu; yapay zekâyâ karşı düşmanca arama.

Basit başlangıç: pizza oylaması

Üç arkadaş tek pizzada anlaşmalı: her biri malzeme önerir, sonra oylarlar. Küçük bir konsol uygulaması bile oylama kuralını ve stratejik oyu gösterir (Bölüm 6).

Kaynaklar

- T. Roughgarden, *Twenty Lectures on Algorithmic Game Theory*, Cambridge University Press, 2016.
- A. Rubinstein, "Perfect Equilibrium in a Bargaining Model", *Econometrica*, 1982.
- E. Calvano, G. Calzolari, V. Denicolò, S. Pastorello, "Artificial Intelligence, Algorithmic Pricing, and Collusion", *American Economic Review*, 2020.
- V. Conitzer, T. Sandholm, "Complexity of Mechanism Design", *UAI*, 2002.
- P. Dütting, Z. Feng, H. Narasimhan, D. Parkes, S. Ravindranath, "Optimal Auctions through Deep Learning", *ICML*, 2019.
- F. McSherry, K. Talwar, "Mechanism Design via Differential Privacy", *FOCS*, 2007.
- R. Cole, T. Roughgarden, "The Sample Complexity of Revenue Maximization", *STOC*, 2014.
- T. Roughgarden, "Transaction Fee Mechanism Design", *ACM EC*, 2021.
- P. Daian ve diğ., "Flash Boys 2.0: Frontrunning in Decentralized Exchanges, Miner Extractable Value, and Consensus Instability", *IEEE S&P*, 2020.
- N. Brown, T. Sandholm, "Superhuman AI for Multiplayer Poker", *Science*, 2019.
- D. Kempe, J. Kleinberg, É. Tardos, "Maximizing the Spread of Influence through a Social Network", *KDD*, 2003.

Dersten Sonra: Kitaplar, Ders Sayfaları, Araçlar

Kitaplar

- N. Nisan, T. Roughgarden, É. Tardos, V. Vazirani (ed.), *Algorithmic Game Theory*, Cambridge UP, 2007.
- A. R. Karlin, Y. Peres, *Game Theory, Alive*, AMS, 2017.
- D. Easley, J. Kleinberg, *Networks, Crowds, and Markets*, Cambridge UP, 2010.
- R. A. Hearn, E. D. Demaine, *Games, Puzzles, and Computation*, A K Peters, 2009.

Ders sayfaları

- [T. Roughgarden, CS364A](#) (Stanford)
- [N. Shah, CSC304](#) (Toronto)
- [M. Dinitz, 601.436/636](#) (Johns Hopkins)
- [A. Procaccia, 15-896](#) (CMU)
- [M. Hoefer, AGT](#) (Frankfurt)

Araçlar

- [Gambit](#): oyun ağacı ve matris çizme, denge hesaplama.
- [Nashpy](#): Python'da iki oyunculu oyunların dengeleri.
- [Game Theory 101](#): video dizisi.
- [React JS oyun örnekleri](#): web arayüzü fikirleri.

Son

Bölüm 14: Gelecek Yönelimler

Dersin sonu · [Çalışma soruları](#) · [Terim sözlüğü](#)