Evolutionary and Metaheuristic Optimization

Guide to AI and Data Courses · Part II: Branches of AI

Population and fitness | Genetic algorithms | Local search | Swarm intelligence | Multi-objective search | Neighboring courses

Sercan KÜLCÜ · Giresun University · Computer Engineering

© Sercan Külcü. All rights reserved. · Interactive HTML version of these slides:
https://sercankulcu.github.io/files/ai_courses/slides/Chapter_10_Evolutionary_and_Metaheuristic_Optimization.html

INTRODUCTION

What Question Does It Ask?

> **How do we find a good solution when there is no gradient or the search space is huge?**

Evolutionary and metaheuristic optimization searches for a good solution to problems without derivatives or with huge search spaces through a loop of **trying, selecting and varying**. Inspired by processes such as natural selection, annealing of metals or ant colonies, these methods do not guarantee the optimum but find a good solution quickly.

At a glance

APPROACH	INPUT	OUTPUT
population-based heuristic search	objective, solution encoding	good, not proven optimal, solution

SUCCESS MEASURE	MATH LOAD	DATA NEEDS
quality, evaluation count	low-medium	none; evaluations suffice

A Short History

1975
Holland publishes "Adaptation in Natural and Artificial Systems", laying the basis of genetic algorithms.

1983
Kirkpatrick, Gelatt and Vecchi introduce simulated annealing.

1992
Dorigo proposes ant colony optimization in his PhD thesis.

1995
Kennedy and Eberhart publish particle swarm optimization (PSO).

1997
Wolpert and Macready publish the "no free lunch" theorem: no method wins on all problems.

2002
Deb and colleagues publish NSGA-II for multi-objective search.

THE COURSE

Core Concepts

Representation
How a solution is encoded: bit string, permutation, real vector.

Fitness
The objective value that says how good a solution is.

Selection
Giving good solutions more offspring: tournament, roulette.

Crossover
Combining parts of two parents into a new solution.

Mutation
Changing a solution by small random steps to add diversity.

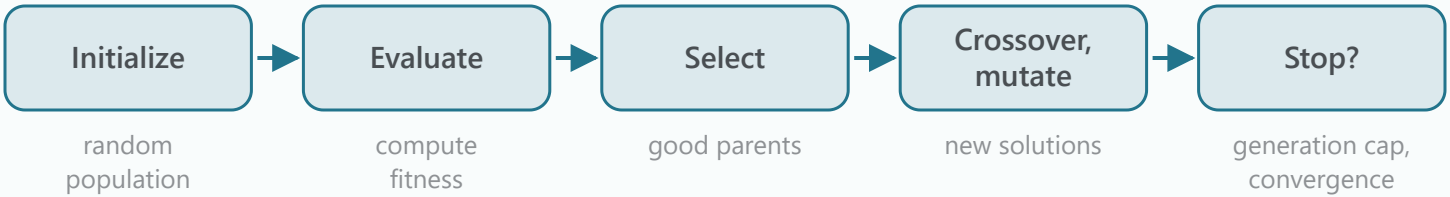
Exploration vs. exploitation
Balance between searching new regions and refining a good one.

Local search
Hill climbing, simulated annealing, tabu search.

Pareto front
Solutions in a multi-objective problem that do not dominate each other.

THE COURSE

A Typical Workflow



Example: A courier company wants the shortest tour through 50 stops. 100 random tours are generated, each tour's length is measured, short tours are selected as parents, their orders are combined and a few stops are swapped. After hundreds of generations a tour close to the best known solution is found.

THE COURSE

What Does the Course Cover?

1. Optimization problems, search spaces, representations
2. Local search: hill climbing, tabu search
3. Simulated annealing and cooling schedules
4. Genetic algorithms: selection, crossover, mutation
5. Evolution strategies and CMA-ES
6. Differential evolution
7. Genetic programming
8. Ant colony optimization
9. Particle swarm optimization
10. Constraint handling: penalties and repair
11. Multi-objective search: Pareto, NSGA-II
12. Experimental design and statistical comparison

Skills you will gain

building evolutionary algorithms with DEAP

multi-objective search with pymoo

hyperparameter search with Optuna

choosing a representation for a problem

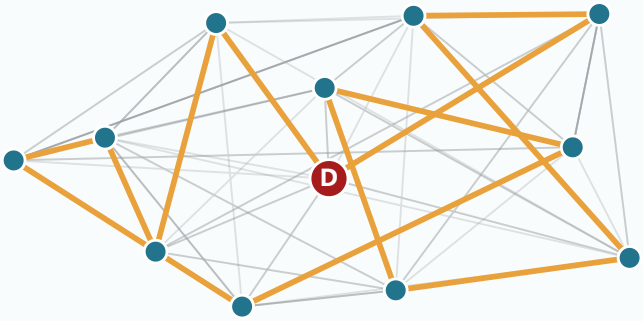
fair comparisons over many runs

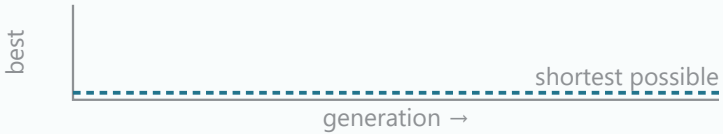
A typical semester; the order and coverage vary between universities.

TRY IT

Evolve the Delivery Van's Route

The delivery van leaves the depot (D), visits 11 addresses and returns. A genetic algorithm improves a population of 40 tours by crossover and mutation. Advance generations, change the mutation rate and compare.





Reset | 1 generation | 50 generations

Mutation rate = **20%**

Generation **0** · best tour **741.6 km**
Initial best 741.6 km · shortest possible 456.0 km
62.6% longer than the shortest. 0 tours tried (about 20 million possible tours).

Set mutation to 0% and reset: the population fills with near-identical tours and the search often **stalls early**. Mutation preserves diversity.

An evolutionary method does not **search for** a solution, it **breeds** one: it combines parts of good tours and keeps novelty with small changes. It does not guarantee the optimum, but quickly finds a good tour in huge search spaces.

CONTEXT

Prerequisites

Courses worth knowing before this one, and why they are needed.

Course	Why it is needed
Artificial Intelligence	heuristic search
Algorithms	complexity, NP-hard problems and greedy methods
Programming	arrays, random number generation and loops

CONTEXT

Are You Ready for This Course?

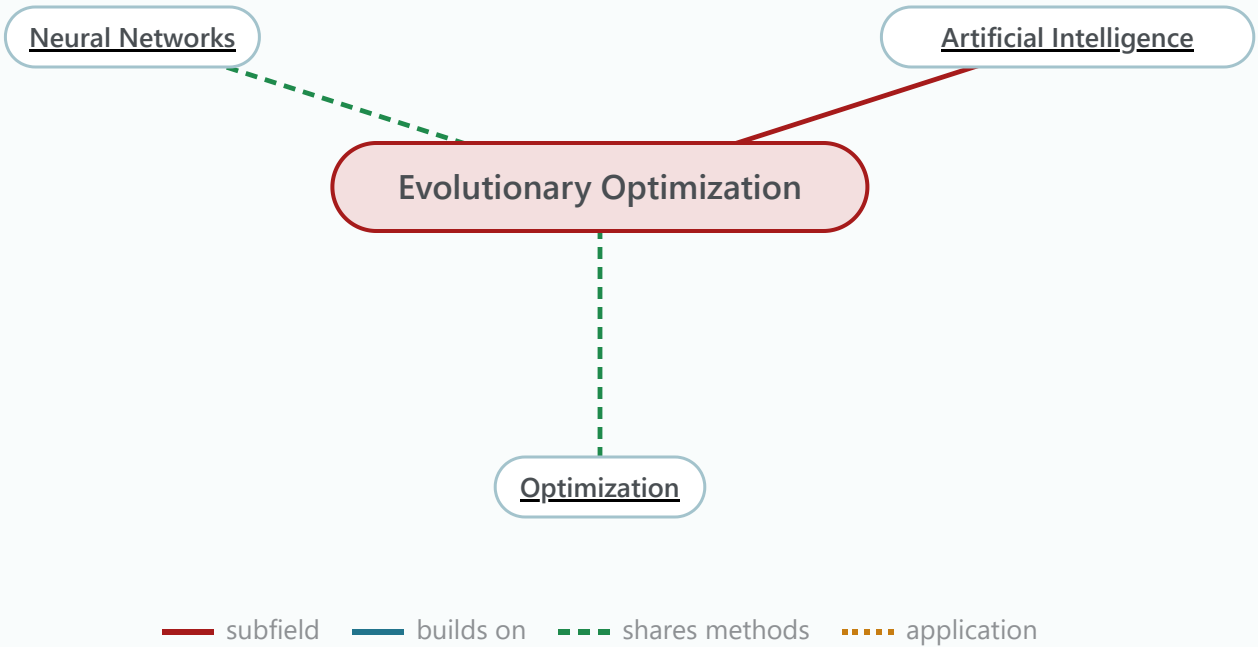
Ask yourself; for every "no", the right-hand column shows where to start.

Ask yourself	If not, start with
Can you state the time complexity of an algorithm in big-O notation?	Complexity and NP-hardness from an algorithms course
Can you find the minimum of a function by taking its derivative?	Optimization
Can you explain mean, standard deviation and a hypothesis test?	Probability and Statistics
Can you write a small Python program with arrays, loops and random numbers?	An intro programming course, then a short NumPy tutorial

If you can say "yes" to most of these, you are ready; one or two gaps can be closed during the semester.

CONTEXT

Neighbouring Courses



OVERLAP

Overlapping Courses

Relation	Course	Shared ground
subfield	This course is a subfield of Artificial Intelligence	heuristic search
shares methods	Shares methods with Artificial Neural Networks	neuroevolution, hyperparameter search
shares methods	Shares methods with Optimization	derivative-free and global optimization

Where It Differs (1/2)

Criterion	Evolutionary Optimization	Optimization	Artificial Intelligence
Core question	How do we find a good solution when there is no gradient or the search space is huge?	How do we find the values that make an objective function best?	How does a machine act intelligently by searching, reasoning and planning?
Approach	population-based heuristic search	mathematical model of objective and constraints	search, logic, probability and learning combined
Input	objective, solution encoding	objective function and constraints	problem definition: states, actions, goal
Output	good, not proven optimal, solution	optimal or near-optimal solution	action sequence, plan or decision
Success measure	quality, evaluation count	objective value, convergence rate	reaching the goal, cost, rationality
Math load	low-medium	the theory itself	medium: logic, probability, graphs
Data needs	none; evaluations suffice	none	little; often a model and knowledge suffice

- **Optimization:** Optimization reaches a provable optimum using derivatives and convexity; this course teaches derivative-free search with no guarantee but wide applicability.
- **Artificial Intelligence:** Searches such as A* in AI find a path to a goal in a state space; this course searches directly for a good complete solution.

Where It Differs (2/2)

Criterion	Evolutionary Optimization	Reinforcement Learning	Neural Networks
Core question	How do we find a good solution when there is no gradient or the search space is huge?	How can good decisions be learned by trial and error from a reward signal alone?	How do networks of simple units learn by adjusting their weights?
Approach	population-based heuristic search	trial and error with rewards	network of simple weighted units
Input	objective, solution encoding	environment: state, reward	numeric feature vector
Output	good, not proven optimal, solution	policy: state → action	class or numeric prediction
Success measure	quality, evaluation count	total expected reward	loss and accuracy
Math load	low-medium	high: probability, dynamic programming	medium-high: derivatives, matrices
Data needs	none; evaluations suffice	generated through interaction	moderate

- **Reinforcement Learning:** RL learns a policy from states to actions by interacting with an environment; this course seeks a good solution for a single objective function.
- **Neural Networks:** Neural networks train weights with gradients; this course searches without gradients, for example evolving network structure or hyperparameters.

PRACTICE

Applications

Vehicle routing
Shortening courier and delivery tours.

Scheduling
Timetables, production and shift plans.

Engineering design
Designing antenna, wing and structural shapes.

Hyperparameter search
Finding settings for learning models.

Layout
Circuit component and facility placement.

Portfolio selection
Multi-objective trade-off between return and risk.

PRACTICE

Risks and Ethics

Loopholes in the fitness function

Search finds and exploits flaws in the objective: the score rises but the intended behaviour never appears; evolution runs often show this.

Unfair comparisons

A single run, unequal evaluation budgets or tuning only your own method make a new algorithm look better than it is.

Designs proven only in simulation

An antenna or structure that looks great in simulation can fail in reality because of effects the model leaves out.

Compute and energy cost

Running expensive simulations over thousands of generations can burn days of processor time and a lot of energy.

In depth: Explainable AI and AI Ethics.

PRACTICE

Tools and Datasets

Tools and libraries

DEAP | pymoo | jMetal | PyGAD | ECJ | MATLAB Global Optimization Toolbox | Optuna

Classic benchmark problems

TSPLIB (traveling salesman) | Knapsack problem | OneMax | Rastrigin and Ackley functions | ZDT and DTLZ multi-objective tests | CEC benchmark functions

This course uses **benchmark problems** instead of datasets: methods are compared on the same problems with the same evaluation budget.

PRACTICE

Common Misconceptions

Common belief	In fact
A genetic algorithm finds the optimal solution.	Metaheuristics do not guarantee the optimum; they aim for a good solution in reasonable time.
Nature-inspired methods are more powerful.	The inspiration does not determine performance; many newly named methods are minor variants of older ones.
One method is best for every problem.	By the "no free lunch" theorem, no method is better on average over all problems.
Use evolutionary methods even when derivatives exist.	When derivatives are available, gradient methods usually reach the result with far fewer evaluations.

NEXT STEPS

After This Course

Courses that build on or extend this one

This course is mostly an end point: it builds on the others.

Related careers

Operations researcher

Logistics and planning engineer

Design optimization engineer

Machine learning engineer

This course goes beyond where classical optimization reaches. Next you can take Optimization for provable methods, Artificial Neural Networks to evolve network structures, or Multi-Agent Systems for an agent view of swarm behavior.

NEXT STEPS

Further Reading

- Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley, 1989.
- Eiben and Smith, *Introduction to Evolutionary Computing*, 2nd ed., Springer, 2015.
- Talbi, *Metaheuristics: From Design to Implementation*, Wiley, 2009.
- Luke, *Essentials of Metaheuristics*, 2nd ed., Lulu, 2013.

The End

Chapter 10: Evolutionary and Metaheuristic Optimization

← Previous chapter · Map of the field · Next chapter →