Metaheuristic Search

Alexander G. Ororbia II

Biologically-Inspired Intelligent Systems

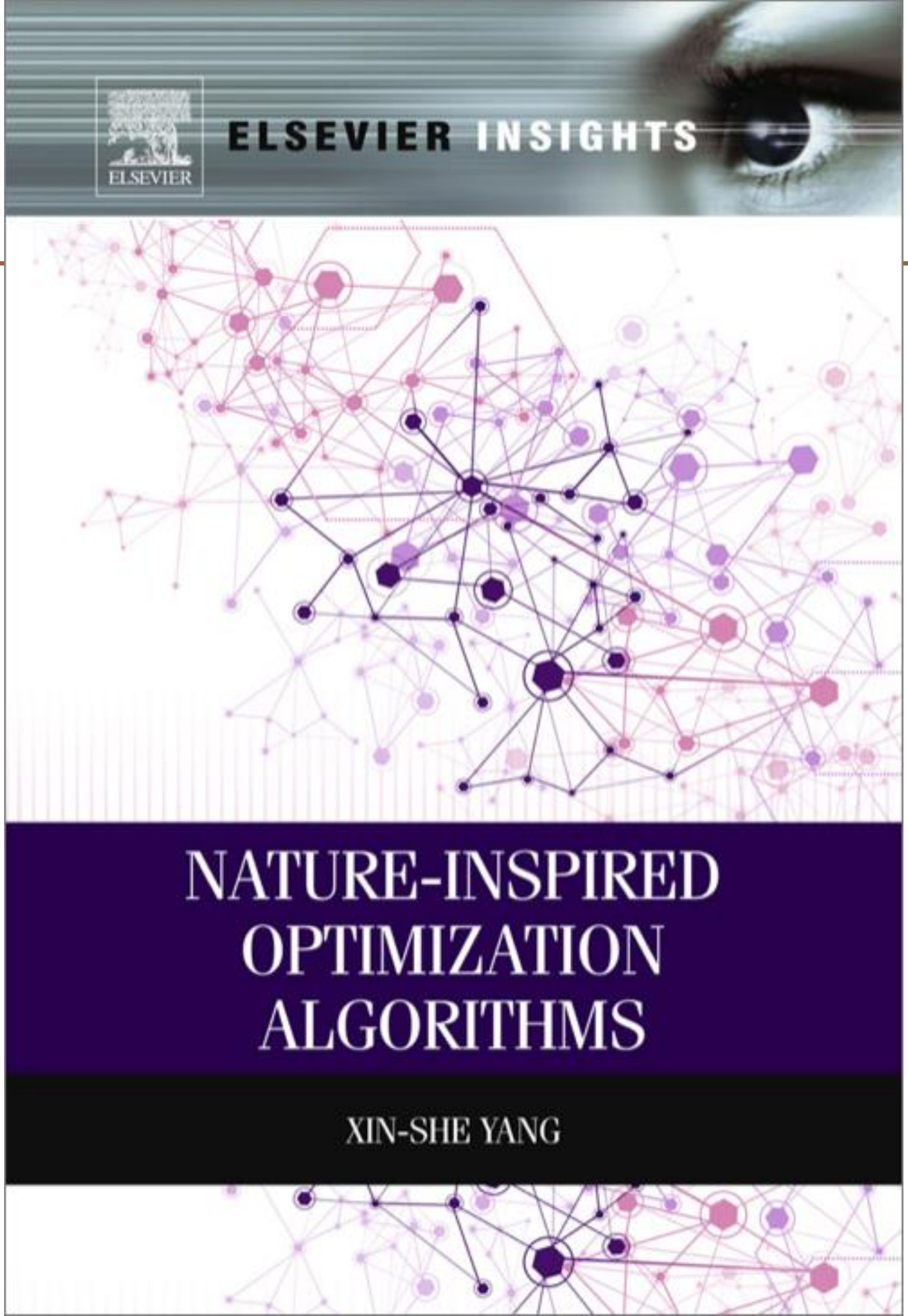
CSCI-633

1/16/2024

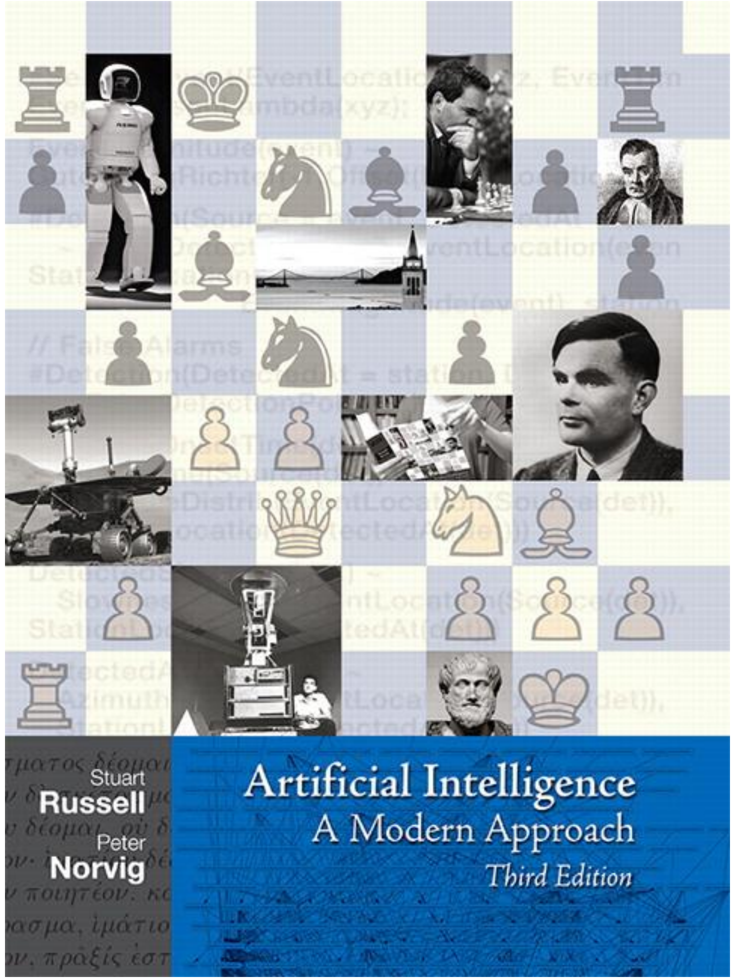
Special thanks to: Rhyd Lewis

Course Page/Syllabus Up

- Syllabus and policy:
- https://www.cs.rit.edu/~ago/courses/633/index.html
- Prerequisites:
- (CSCI-603 and CSCI-605 and CSCI-661 with grades of B or better) or ((CSCI-243 or SWEN-262) and (CSCI-262 or CSCI-263)) or equivalent courses


ELSEVIER
ELSEVIER INSIGHTS
NATURE-INSPIRED OPTIMIZATION ALGORITHMS
XIN-SHE YANG

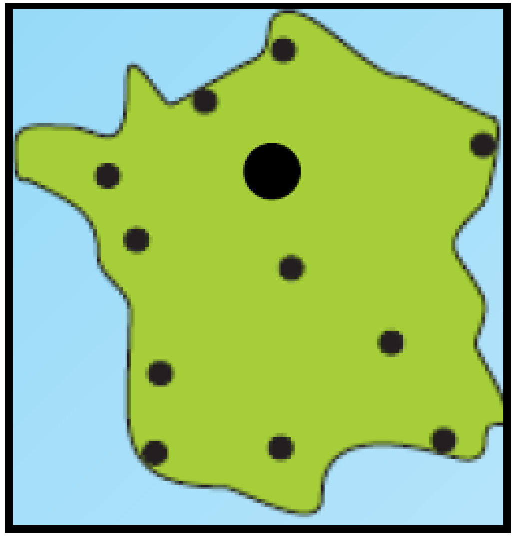
From 630!!


Stuart
Russell
Peter
Norvig
Artificial Intelligence
A Modern Approach
Third Edition

The Traveling Salesman (TSP) Problem

- A classic combinatorial optimisation problem
- Given n cities on a map, find the shortest route that visits all cities once, and starts and ends at the same city.

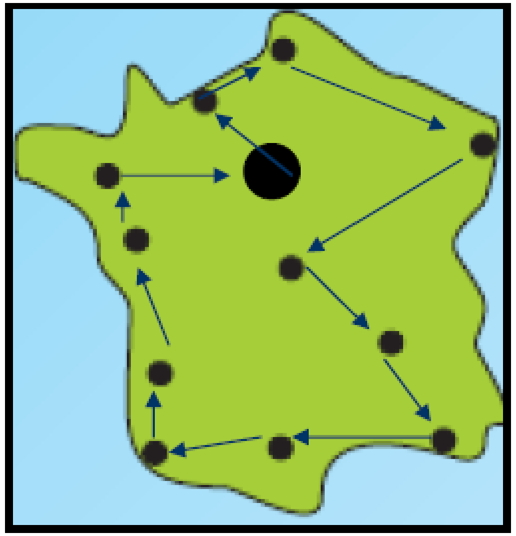
Starting and ending in Paris, which route allows us to visit all major cities with the least amount of travelling?
(We assume straight-line distances between cities for now)



The Traveling Salesman (TSP) Problem

- A classic combinatorial optimisation problem
- Given n cities on a map, find the shortest route that visits all cities once, and starts and ends at the same city.

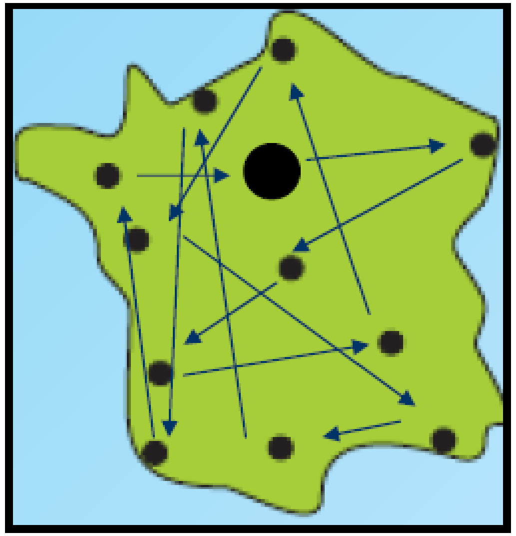
Starting and ending in Paris, which route allows us to visit all major cities with the least amount of travelling?
(We assume straight-line distances between cities for now)



The Traveling Salesman (TSP) Problem

- A classic combinatorial optimisation problem
- Given n cities on a map, find the shortest route that visits all cities once, and starts and ends at the same city.

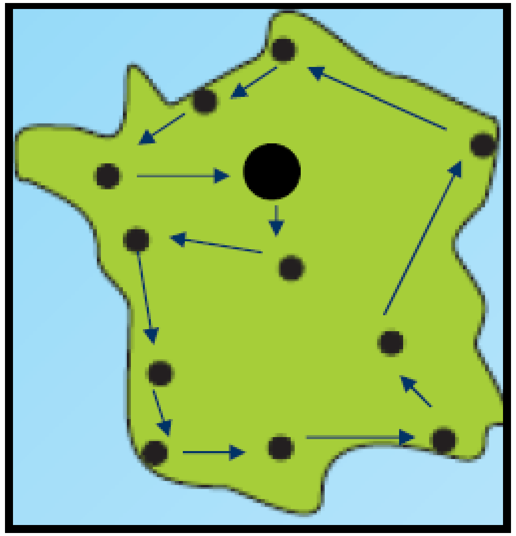
Starting and ending in Paris, which route allows us to visit all major cities with the least amount of travelling?
(We assume straight-line distances between cities for now)



The Traveling Salesman (TSP) Problem

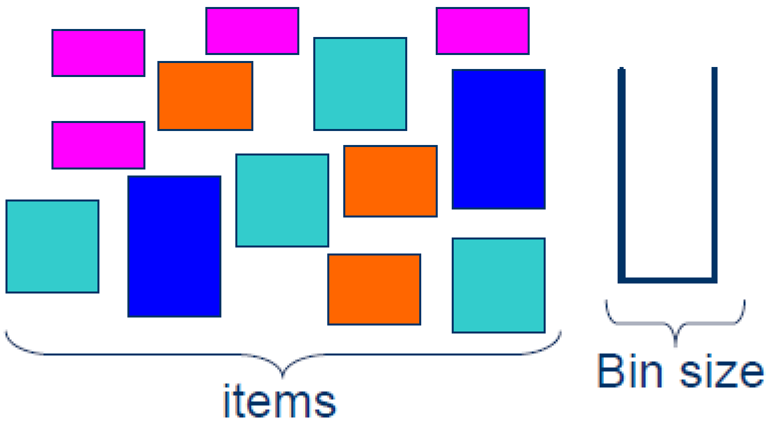
- A classic combinatorial optimisation problem
- Given n cities on a map, find the shortest route that visits all cities once, and starts and ends at the same city.

Starting and ending in Paris, which route allows us to visit all major cities with the least amount of travelling?
(We assume straight-line distances between cities for now)



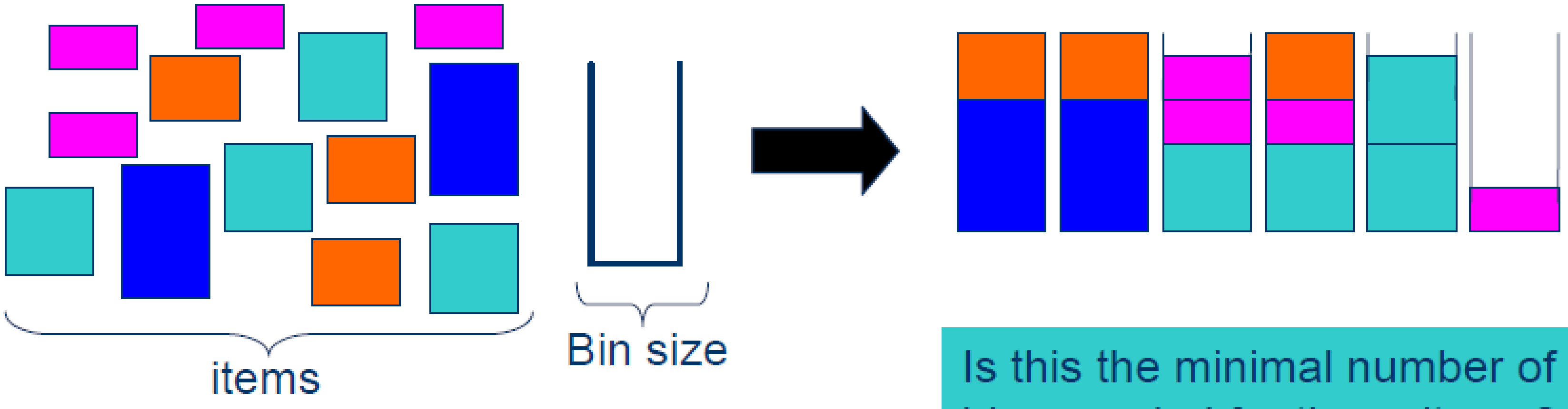
The 1-Dimensional Bin Packing Problem

Given n items of different (1D) sizes, and given some fixed-capacity bins, pack the items into a **minimum** number of bins



The 1-Dimensional Bin Packing Problem

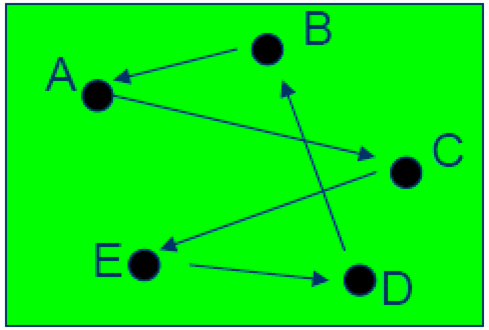
Given n items of different (1D) sizes, and given some fixed-capacity bins, pack the items into a **minimum** number of bins



Is this the minimal number of bins needed for these items?

TSP Growth Rates

- A route around a map can be represented as a permutation of the n cites:
  - E.g. For 5 cities, **[B, A, C, E, D]** means “start at city B, then go to city A, then city C, then E, then D, and return to city B”
- Given n cities, there is a total of $n!$ permutations (where $n! = n \times (n-1) \times (n-2) \times \ldots \times 2 \times 1$)
- Some permutations represent the same routes – there are actually $\frac{1}{2}(n-1)!$ different routes in total.



An Inconvenient Truth

- The only algorithm that will **guarantee** to return the provably optimal solution to **any** instance of the TSP needs to check the majority of – if not all – possible routes.
- However, the number of routes grows exponentially, quickly making the problem intractable:

Number of routes for different n's

Cities (n)	5	10	50	100	1000
Routes [$\frac{1}{2}(n-1)!$]	12	181,440	3.04×10^{62}	4.67×10^{155}	**Lots!!!**

Beyond the TSP…Other Intractable Problems

- Intractable problems arise in many areas:
  - Packing Problems
  - Games (Sudoku, *Tetris*, Minesweeper)
  - Vehicle routing problems
  - Scheduling and Timetabling Problems (see later)
  - Graph theoretic problems, and so on.
- To tackle them, we might:
  - Attempt to avoid or redefine the problem
  - Use some brute-force algorithm and limit ourselves only to small instances
  - Use **approximation algorithms** that will hopefully give us a solution that is “good enough” for practical purposes

Solving Intractable Problems w/ Metaheuristics

- A metaheuristic is a *general algorithmic framework* for addressing intractable problems
- They are often (though not necessarily) inspired by processes occurring in nature, e.g.
  - **Darwinian Natural Selection**
  - Annealing
  - Collective behaviour of ants
- Others merely provide neat ways of exploring the huge search spaces in efficient and effective ways.
- Typically, metaheuristics are approximation algorithms – they cannot always produce provably optimal solutions, but they do have the potential to produce good solutions in short amounts of time (if used appropriately).

Meta- Greek word for upper level methods
Heuristic – Greek word *heuriskein* – art of discovering new strategies to solve problems.

Metaheuristics

The idea: search the solution space directly. No math models, only a set of algorithmic steps, iterative method. Find a feasible solution and improve it. A greedy solution may be a good starting point.

Goal: Find the best solution for a given stopping criteria.

Applied to combinatorial and constraint optimization problems

Diversification and *intensification* of the search are the two strategies for search in metaheuristics.

- Strike *balance* between them -- too much of either yields poor solutions
- Only a limited amount of time to search and are looking for good quality solution (*quality* vs. *time* tradeoff)

Metaheuristic Categorization

Nature inspired (swarm intelligence, biology) vs non-nature inspired (simulated annealing, physics)

Memory usage (tabu search) vs memoryless methods (local search, simulated annealing SA)

Deterministic (tabu, local search) vs stochastic (GA, SA) metaheuristics

- Deterministic – same initial solution leads to same final solution after several search steps
- Stochastic – same initial solution leads to different final solutions due to some randomness in algorithm

Population based search

- Manipulates a whole population of solutions – *exploration* / diversification

Single solution based search

- Manipulates a single solution – *exploitation* / intensification

Iterative vs. Greedy:

- *Iterative* – start w/ complete solution(s) & transform at each iteration
- *Greedy*- start with empty solution, add decision variables until complete solution obtained

Metaheuristic Categorization

Nature inspired (swarm intelligence [illegible] n-nature inspired (simulated annealing, physics)

Memory usage (tabu search) vs [illegible] thods (local search, simulated annealing SA)

Deterministic (tabu, local se[illegible] (GA, SA) metaheuristics

- Deterministic - [illegible] to same final solution after several search steps
- Stochastic – sa[illegible] to different final solutions due to some randomness in algorithm

Populati[illegible]

- Manipulates a whole population of solutions – exploration/diversification

Single solution based search

- Manipulates a single solution – exploitation/intensification

Iterative vs Greedy:

- Iterative – start w/ complete solution(s) & transform at each iteration
- Greedy- start with empty solution, add decision variables until complete solution obtained

Metaheuristic Categorization

Nature inspired (swarm intelligence n-nature inspired (simulated annealing, physics)

Memory usage (tabu search) vs thods (local search, simulated annealing SA)

Deterministic (tabu, local se (GA, SA) metaheuristics

- Deterministic - to same final solution after several search steps
- Stochastic – s to different final solutions due to some randomness in algorithm

Populatio

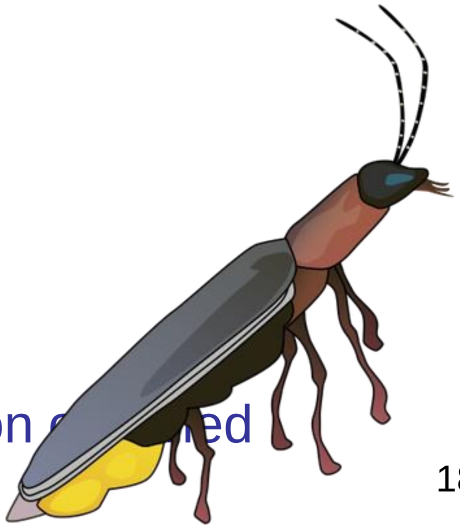
- Manipulates a whole population solutions – exploration/diversification

Single solution based search

- Manipulates a single solution – exploitation/intensification

Iterative vs Greedy:

- Iterative – start w/ complete solution(s) & transform at each iteration
- Greedy- start with empty solution, add decision variables until complete solution ed



Metaheuristic Categorization

Nature inspired (swarm intelligence … n-nature inspired (simulated annealing, physics)

Memory usage (tabu search) vs … thods (local search, simulated annealing SA)

Deterministic (tabu, local se… (GA, SA) metaheuristics

- Deterministic - … to same final solution after several search steps
- Stochastic – s… to different final solutions due to some randomness in algorithm

Population …

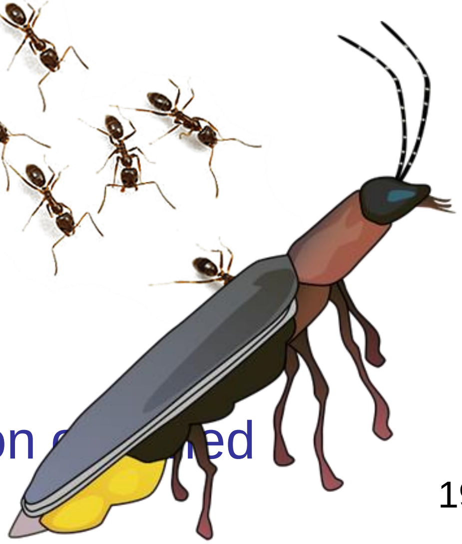
- Manipulates a whole population of solutions – exploration/diversification

Single solution based search

- Manipulates a single solution – exploitation/intensification

Iterative vs Greedy:

- Iterative – start w/ complete solution(s) & transform at each iteration
- Greedy- start with empty solution, add decision variables until complete solution … ed



Metaheuristic Categorization

Nature inspired (swarm intelligence n-nature inspired (si hysics)

Memory usage (tabu search) vs hods (local search,

Deterministic (tabu, local se (GA, SA) metaheuri

- Deterministic - to same final solu on afte
- Stochastic – s to different final so

Populatio

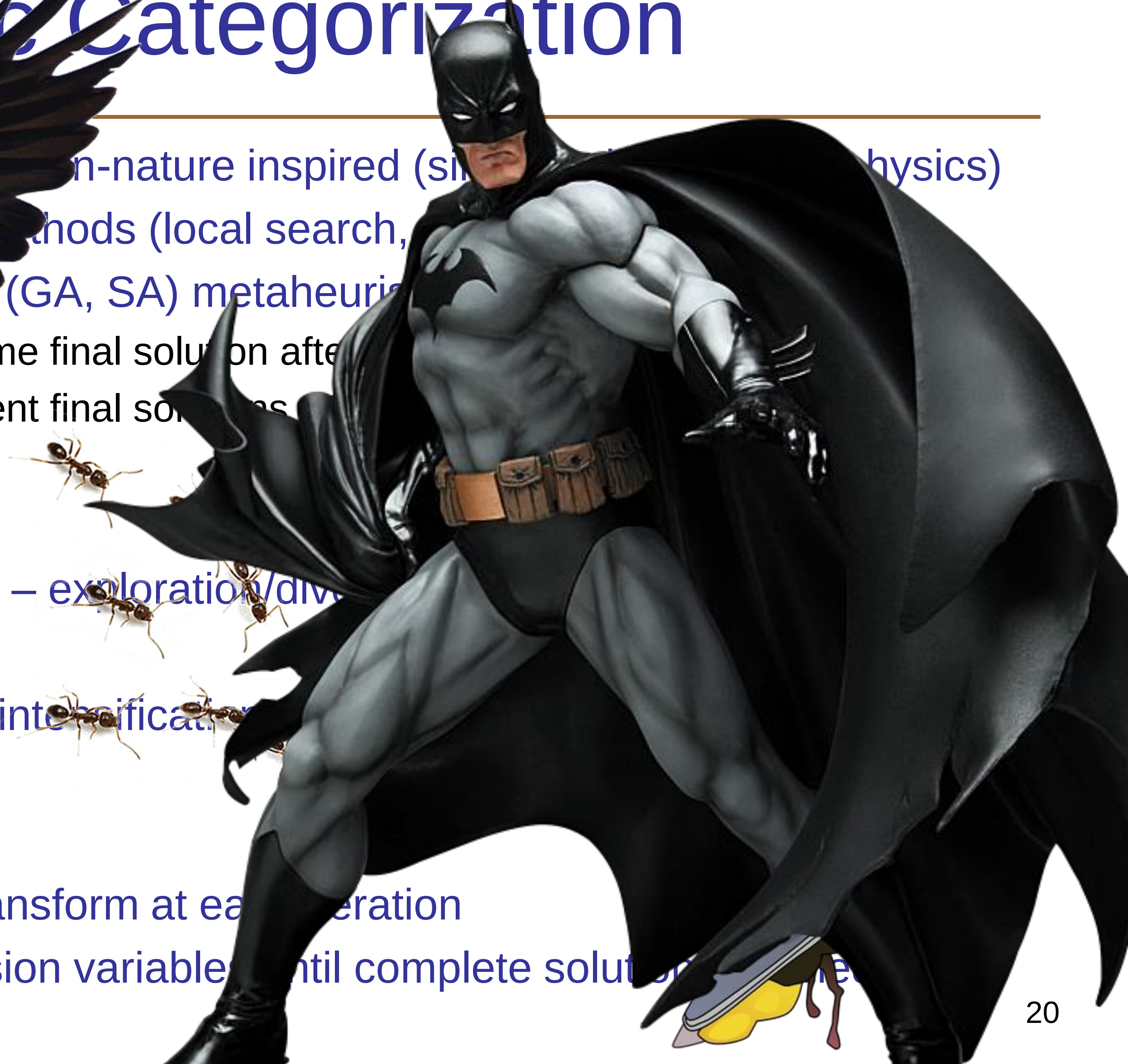
- Manipulates a whole population solutions – exploration/dive

Single solution based search

- Manipulates a single solution – exploitation/inte

Iterative vs Greedy:

- Iterative – start w/ complete solution(s) & transform at ea eration
- Greedy- start with empty solution, add decision variable ntil complete solut



When to use Metaheuristics

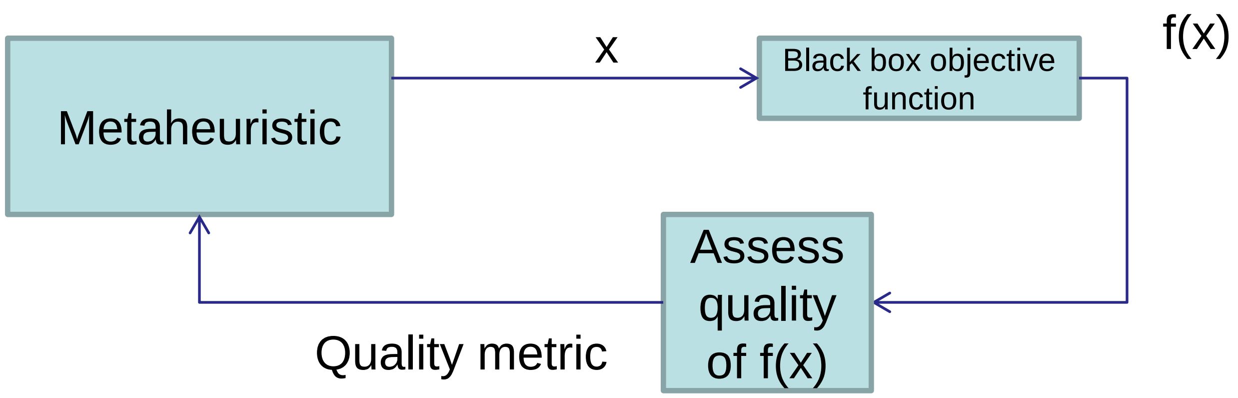
If one can use exact methods then do ***not*** use metaheuristics

P(olynomial) class problem with large number of solutions. P-time algorithms are known but too expensive to implement

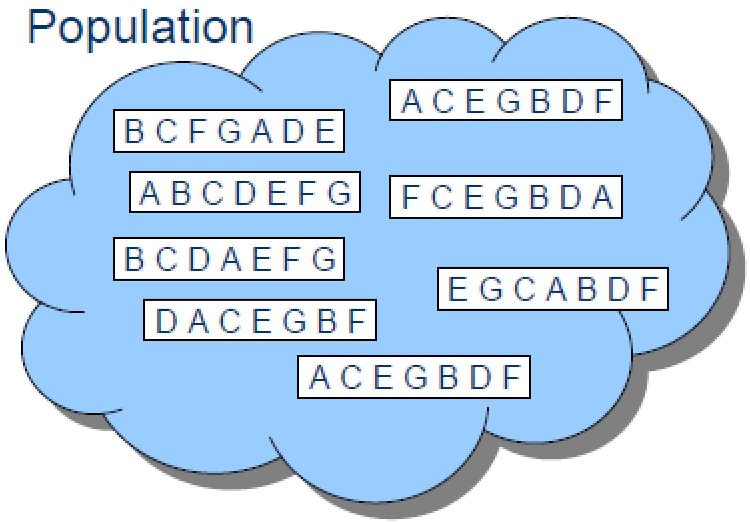
Dynamic optimization - real-time optimization - metaheuristics can reduce search time and still find "good enough" solutions.

A difficult NP-hard problem - even a moderate size problem

Problems where objective function is black box, i.e. often simulated and have no/inaccurate mathematical formulation for objective function(s)

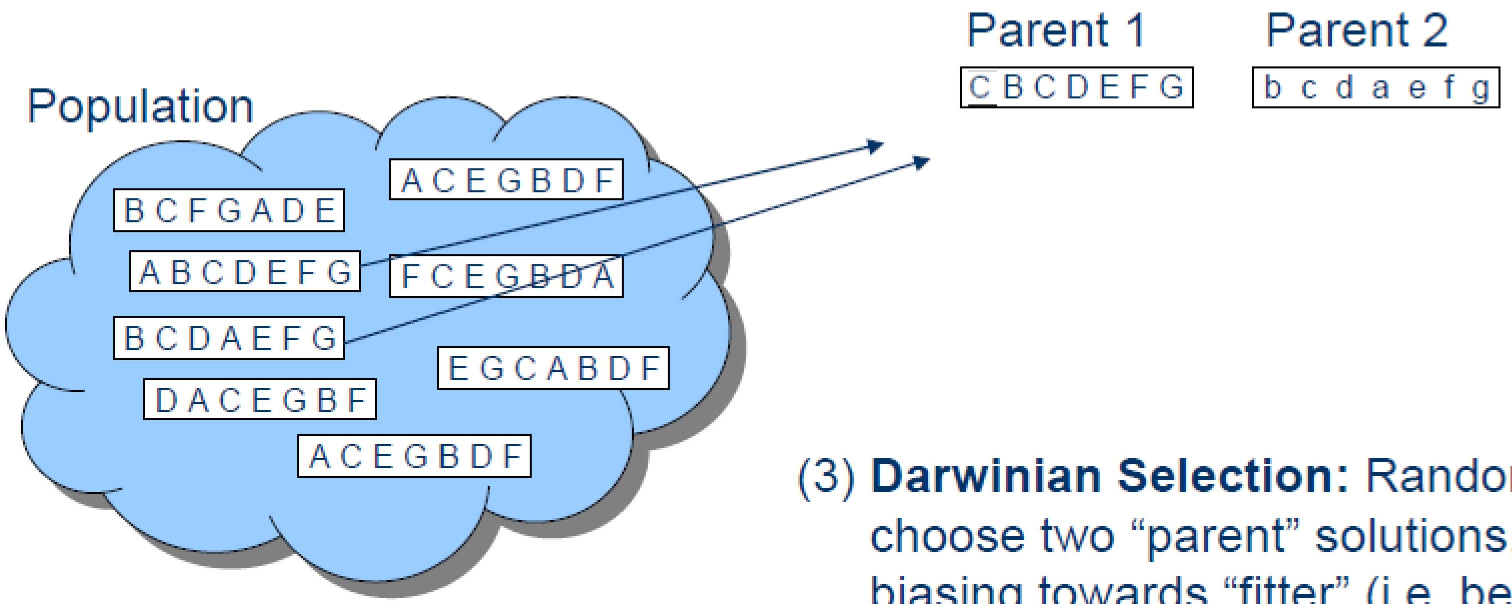


Evolving Solutions for TSP



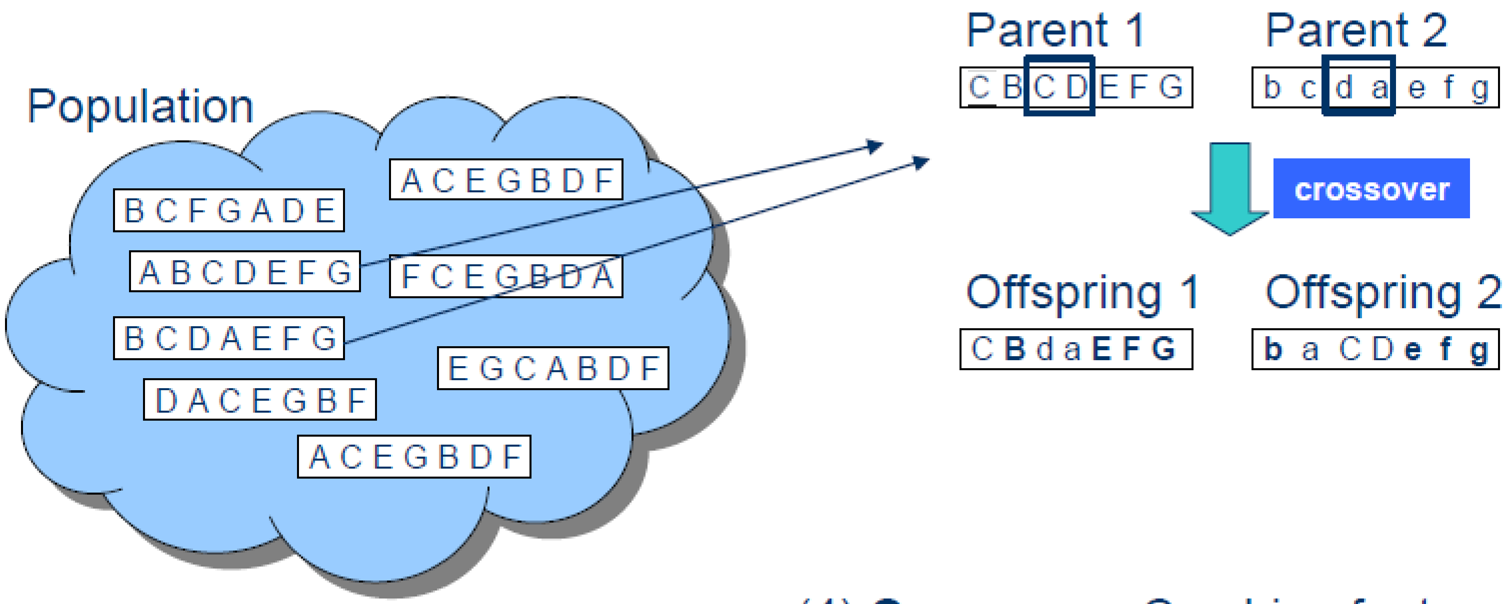
(1) Randomly produce an “**initial population**” of valid candidate solutions.
(2) Calculate the cost **(fitness)** of each member of the population

Evolving Solutions for TSP



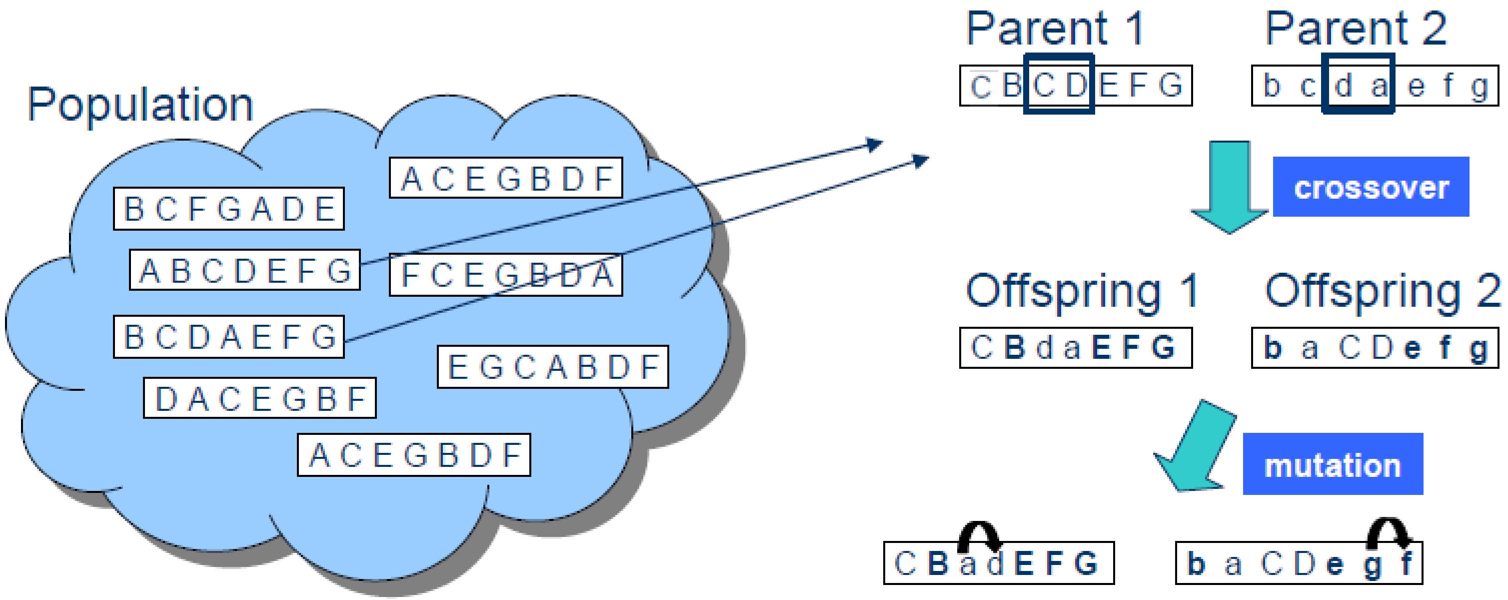
(3) **Darwinian Selection:** Randomly choose two "parent" solutions, biasing towards "fitter" (i.e. better quality solutions)

Evolving Solutions for TSP



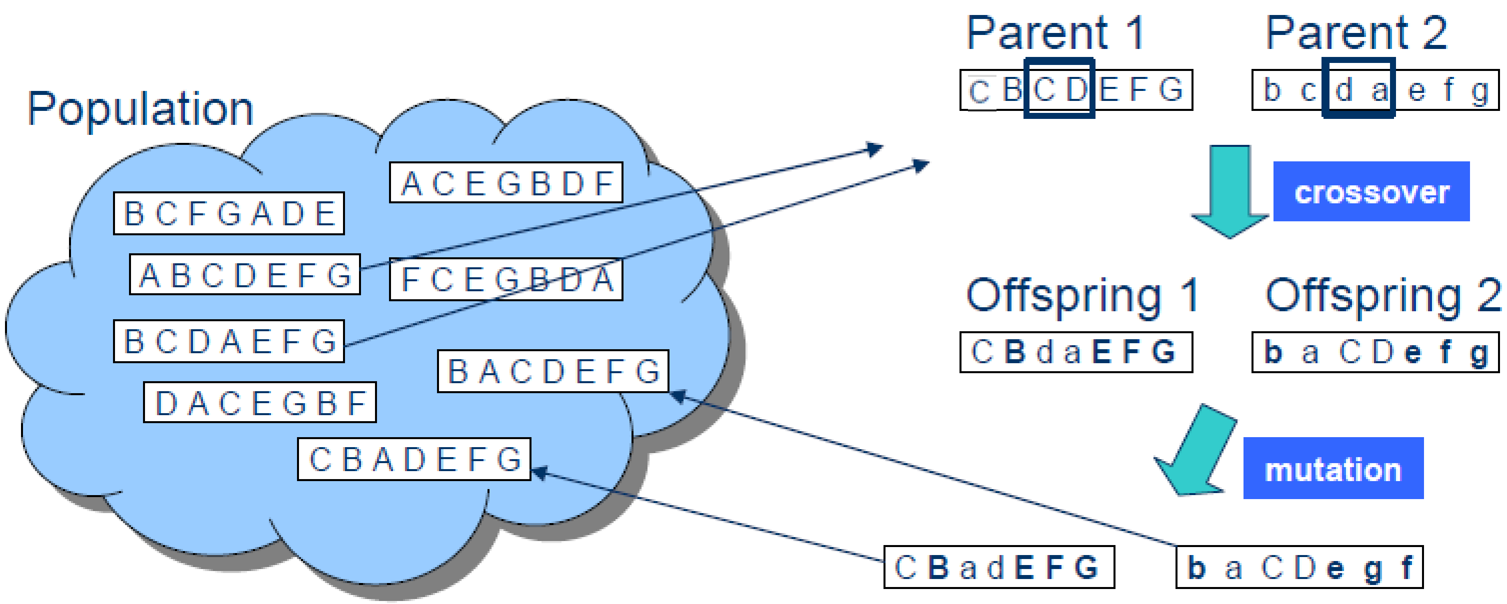
(4) **Crossover:** Combine features of the two parents to form two new "offspring" solutions

Evolving Solutions for TSP



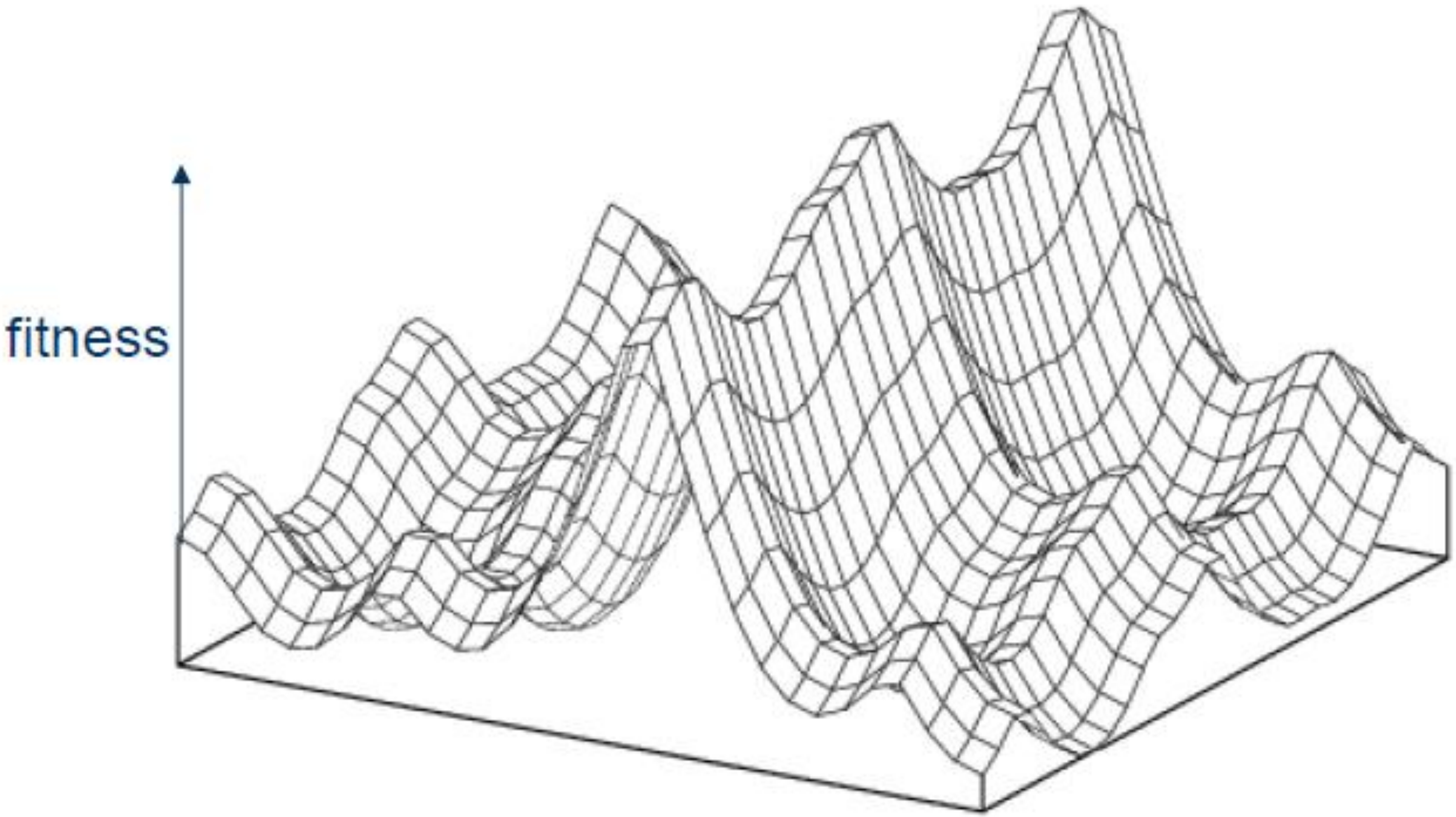
(5) **Mutation:** Make a small number of random alterations to the offspring

Evolving Solutions for TSP



(6) **Replacement:** Reinsert the new offspring back into the population, and go back to Step (3)

Combinatorial Problems: Fitness Landscape



Questions?