Class
Artificial Intelligence

Ari Barrera, Ph.D.

March 2023

UNIVERSIDAD
Panamericana

Local search algorithms and Optimisation Problems

Hill Climbing search (greedy local search)

Simulated Annealing

Genetic Algorithms

Local search algorithms and Optimisation Problems

UNIVERSIDAD Panamericana

Genetic Algorithms

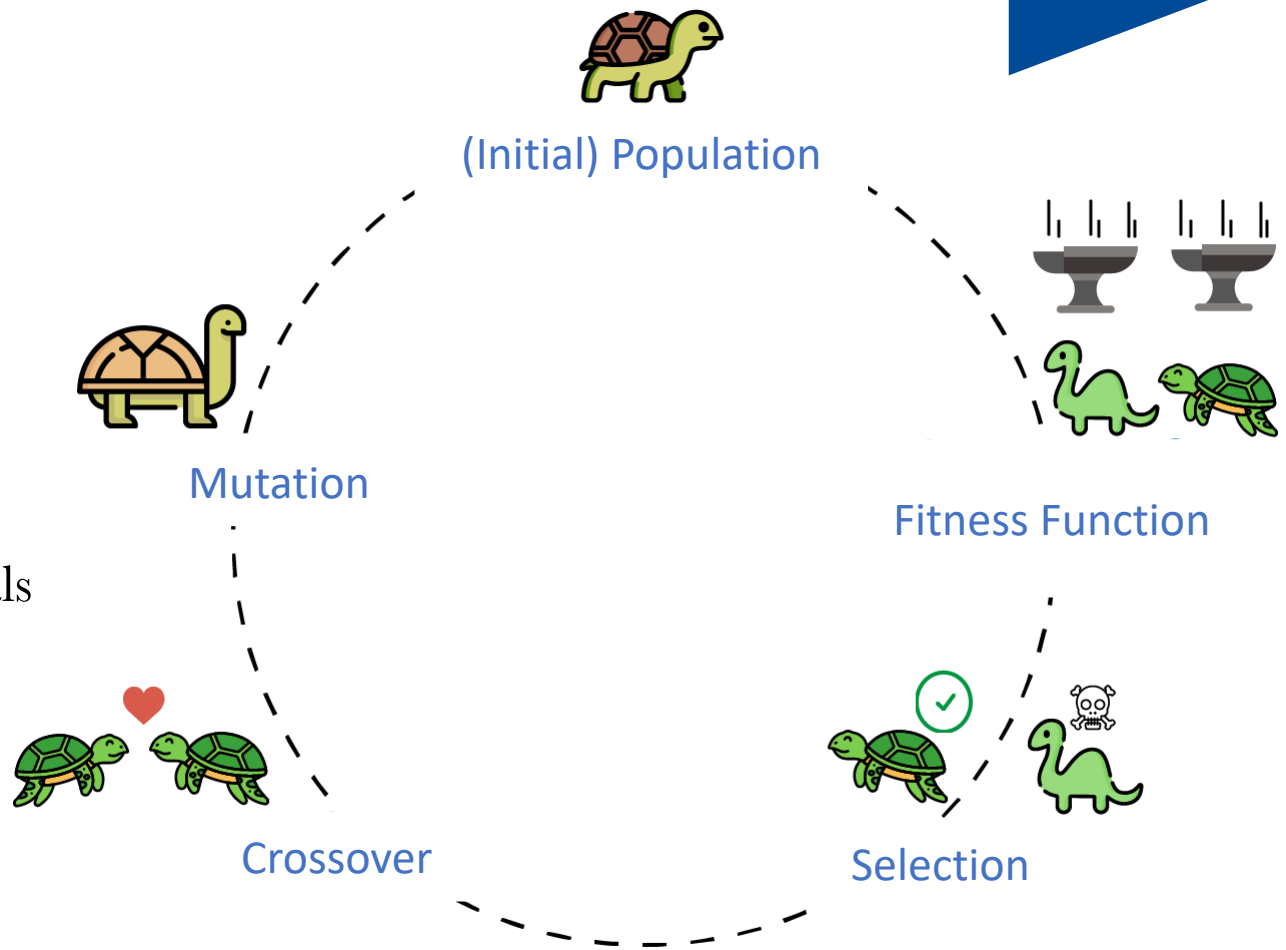
- A third type of metaheuristic that is quite different from the first two

- This type tends to be particularly effective at exploring various parts of the feasible region and gradually evolving toward the best feasible solutions

Local search algorithms and Optimisation Problems

Genetic Algorithms

- Motivated by the metaphor of natural selection in biology:
  - there is a population of individuals (states),
  - in which the fittest (highest value) individuals
  - produce offspring (successor states)
  - that populate the next generation, a process called recombination

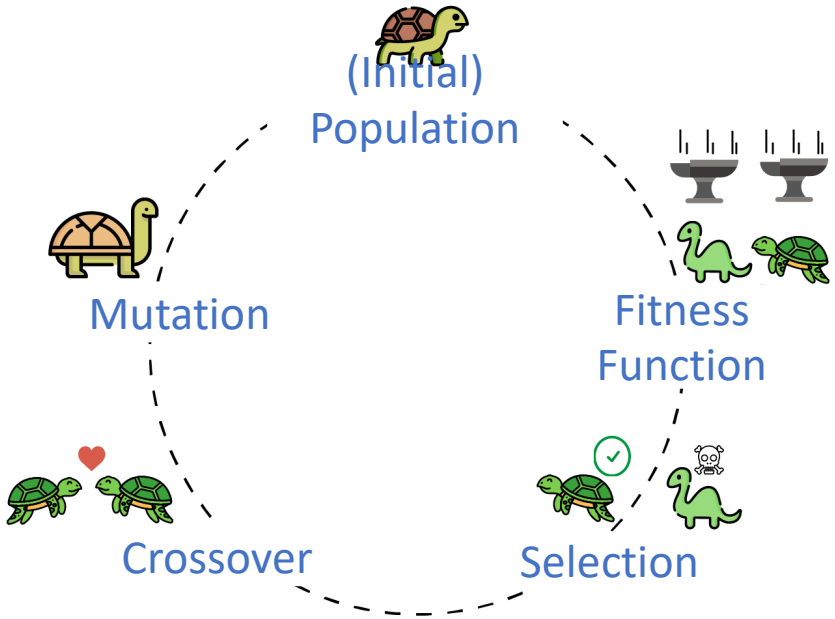


https://www.generativedesign.org/02-deeper-dive/02-04_genetic-algorithms/02-04-01_what-is-a-genetic-algorithm

Local search algorithms and Optimisation Problems

Genetic Algorithms

- The current population consists of the set of trial solutions currently under consideration
- Some of the youngest members of the population (including especially the fittest members) survive into adulthood
- and become parents (paired at random) who then have children (new trial solutions) who share some of the features (genes) of both parents



https://www.generativedesign.org/02-deeper-dive/02-04_genetic-algorithms/02-04-01_what-is-a-genetic-algorithm

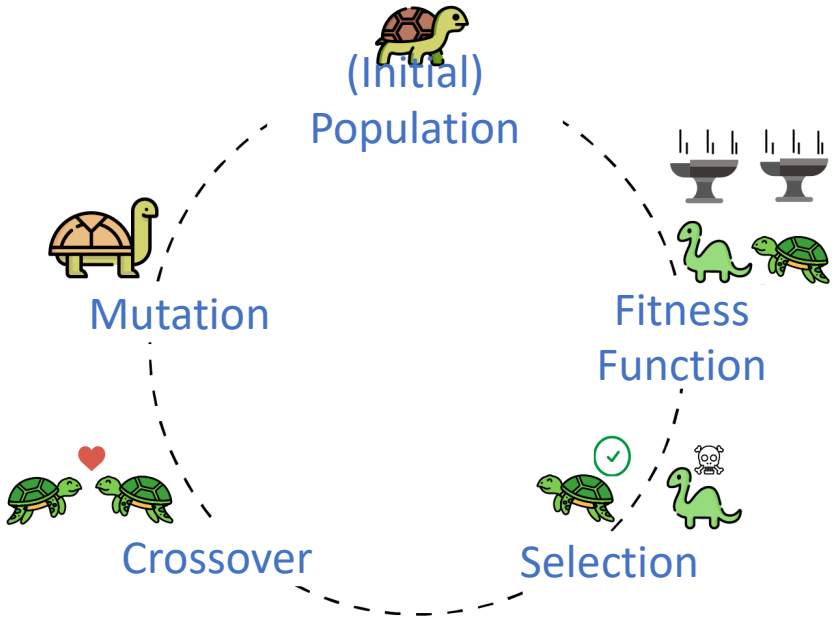
Local search algorithms and Optimisation Problems

Genetic Algorithms

- Since the fittest members of the population are more likely to become parents than others, a genetic algorithm tends to generate improving populations of trial solutions as it proceeds.

- Mutations occasionally occur so that certain children also can acquire features (sometimes desirable features) that are not possessed by either parent.

- This helps a genetic algorithm to explore a new, perhaps better part of the feasible region than previously considered



https://www.generativedesign.org/02-deeper-dive/02-04_genetic-algorithms/02-04-01_what-is-a-genetic-algorithm

Local search algorithms and Optimisation Problems

UNIVERSIDAD Panamericana

Genetic Algorithms

- Although the analogy of the process of biological evolution defines the core of any genetic algorithm, it is not necessary to adhere rigidly to this analogy in every detail

- For example, some genetic algorithms allow the same trial solution to be a parent repeatedly over multiple generations (iterations)

Local search algorithms and Optimisation Problems

Genetic Algorithms – Basic algorithm

- Initialisation
  - Start with an initial population of feasible trial solutions, perhaps by generating them randomly
  - Evaluate the fitness (the value of the objective function) for each member of this current population

Local search algorithms and Optimisation Problems

Genetic Algorithms – Basic algorithm

- Iteration
  - Use a random process that is biased toward the more fit members of the current population to select some of the members (an even number) to become parents
  - Pair up the parents randomly and then have each pair of parents give birth to two children (new fea- sible trial solutions) whose features (genes) are a random mixture of the features of the parents, except for occasional mutations

Local search algorithms and Optimisation Problems

Genetic Algorithms – Basic algorithm

- Iteration
  - Whenever the random mixture of features and any mutations result in an infeasible solution, this is a miscarriage, so the process of attempting to give birth then is repeated until a child is born that corresponds to a feasible solution
  - Retain the children and enough of the best members of the current population to form the new population of the same size for the next iteration

Local search algorithms and Optimisation Problems

Genetic Algorithms – Basic algorithm

- Iteration
  - Discard the other members of the current population
  - Evaluate the fitness for each new member (the children) in the new population

Local search algorithms and Optimisation Problems

Genetic Algorithms – Basic algorithm

- Stopping rule
  - Use some stopping rule, such as a fixed number of iterations, a fixed amount of CPU time, or a fixed number of consecutive iterations without any improvement in the best trial solution found so far
  - Use the best trial solution found on any iteration as the final solution

Local search algorithms and Optimisation Problems

UNIVERSIDAD Panamericana

Genetic Algorithms – Basic algorithm

- Considerations
  - What should the population size be?
  - How should the members of the current population be selected to become parents?
  - How should the features of the children be derived from the features of the parents?
  - How should mutations be injected into the features of the children?
  - Which stopping rule should be used?

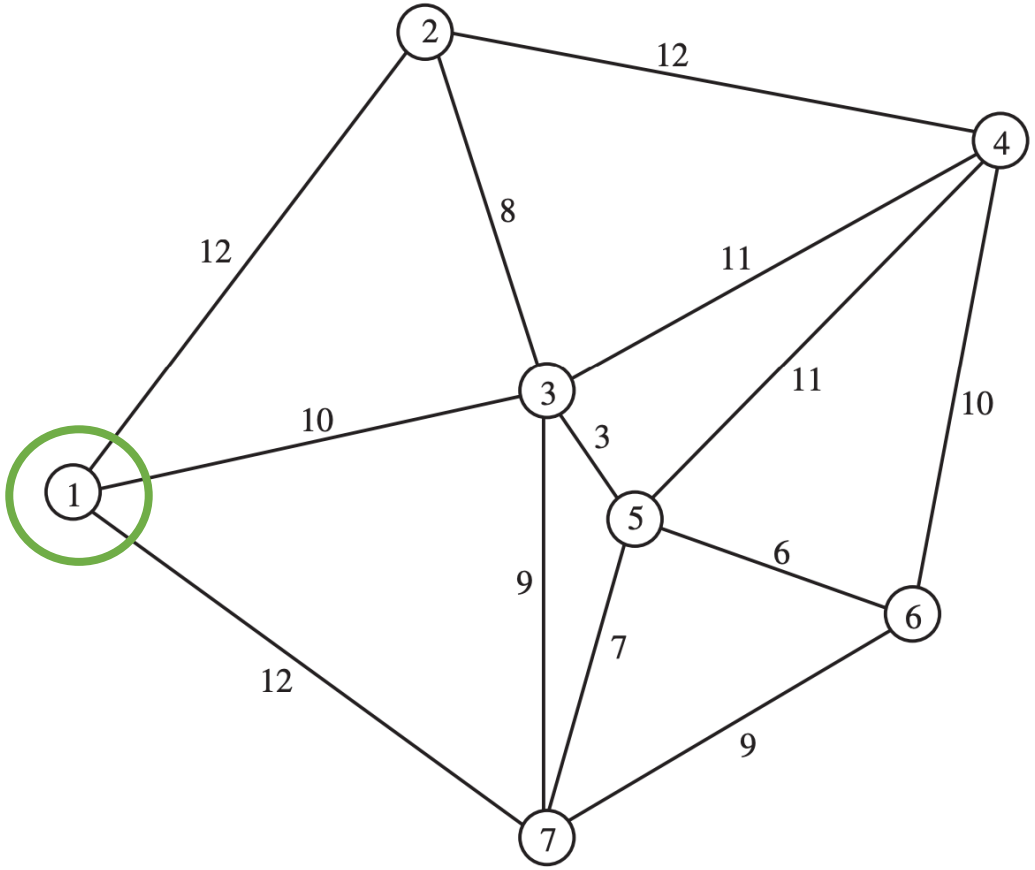
Local search algorithms and Optimisation Problems

Genetic Algorithms – Basic algorithm

- Considerations
  - Population size = 10 (good for small problems)
  - Selection of parents = Random 4 of the 5 fittest members of the population (based on the value of the objective function); and Random 2 of the 5 least fit members. Pair up the six parents randomly to form three couples.
  - Passage of features (genes) from parents to children = This process is highly problem dependent and so it differs. It will be described later
  - Mutations rate = A probability of 0.1 that an inherited feature of a child mutates into an opposite feature (Much smaller mutation rates commonly are used for large problems)
  - Stopping rule = Stop after five consecutive iterations without any improvement

Local search algorithms and Optimisation Problems

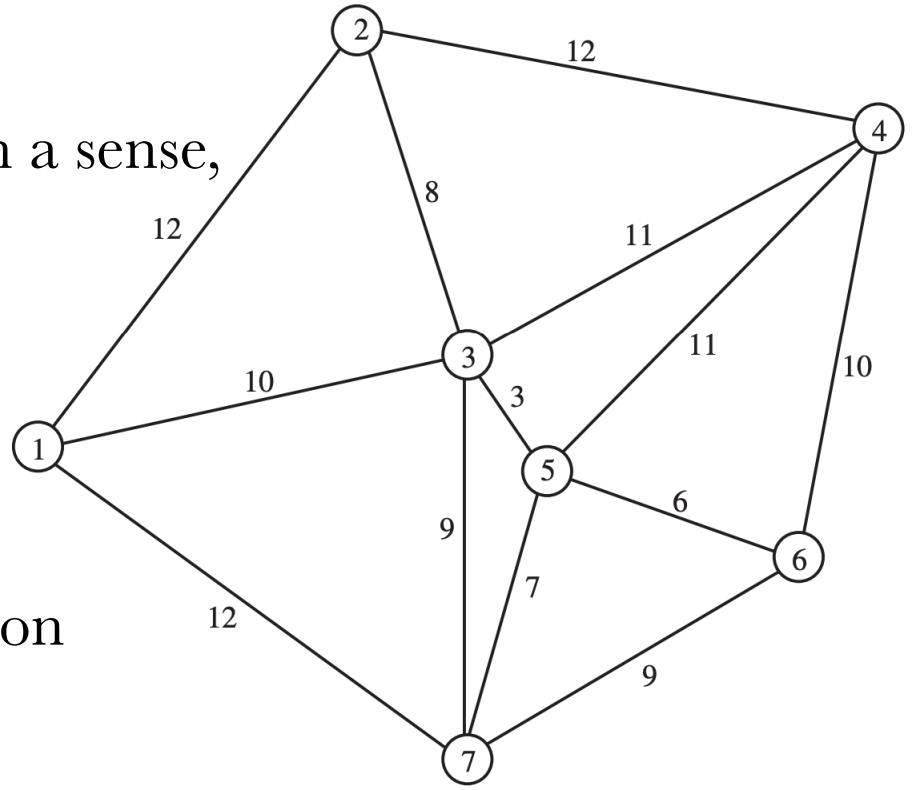
Genetic Algorithms – Basic algorithm – TSP example



Local search algorithms and Optimisation Problems

Genetic Algorithms – Basic algorithm

- A complication with this particular example is that, in a sense, it is too easy
- This problem barely has 10 distinct feasible solutions
- Therefore, it is not possible to have an initial population with 10 distinct trial solutions



Local search algorithms and Optimisation Problems

Genetic Algorithms – Basic algorithm – TSP example

- Fortunately, a genetic algorithm can still operate reasonably well when there is a modest amount of duplication in the trial solutions in a population or in two consecutive populations

- It does not do anything to avoid duplication in the trial solutions considered

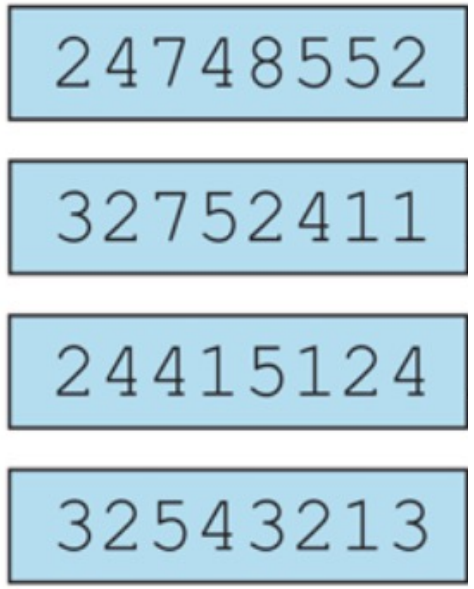
Local search algorithms and Optimisation Problems

Genetic Algorithms – Basic algorithm – TSP example

- First solution = 1-2-3-4-5-6- 7-1*

* encoding solutions

In general, clever methods of representing solutions (often by using strings of binary digits) can make it easier to generate children, create mutations, maintain feasibility, and so forth, in a natural way

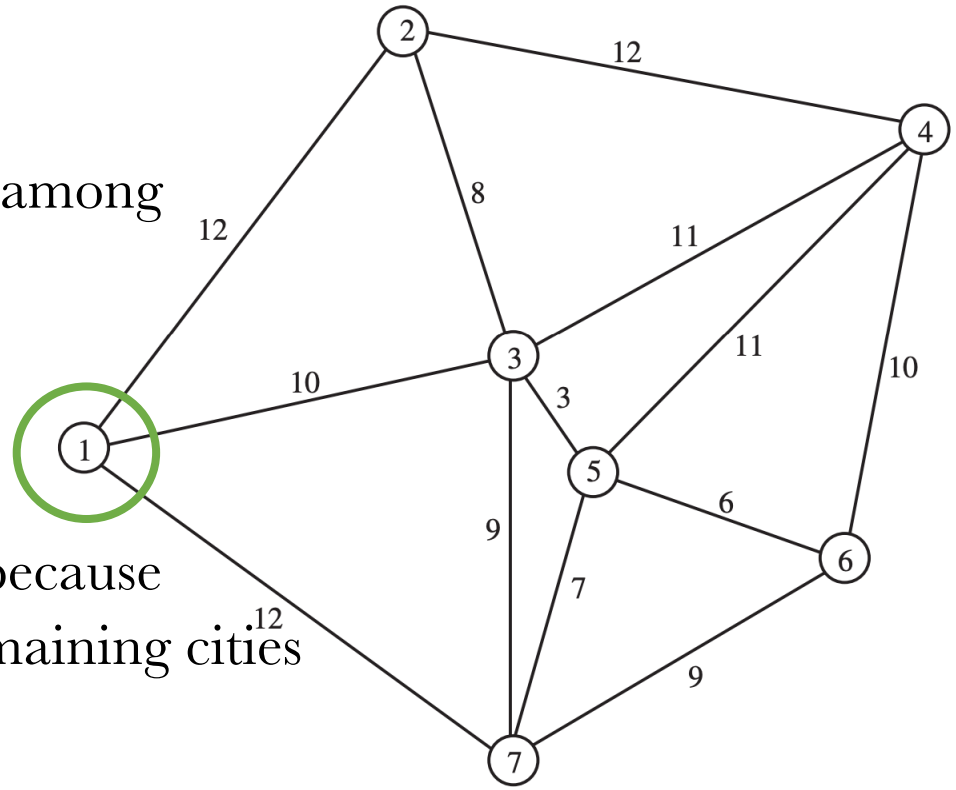


Local search algorithms and Optimisation Problems

UNIVERSIDAD Panamericana

Genetic Algorithms – Basic algorithm – TSP example

- 10 trial solutions for the initial population
- Generate random numbers to select the next city from among those that have a link
- This process is continued until either every city is included once in the tour (plus a return to the home base city from the last city) or a dead end is reached because there is no link from the current city to any of the remaining cities that still need to be visited

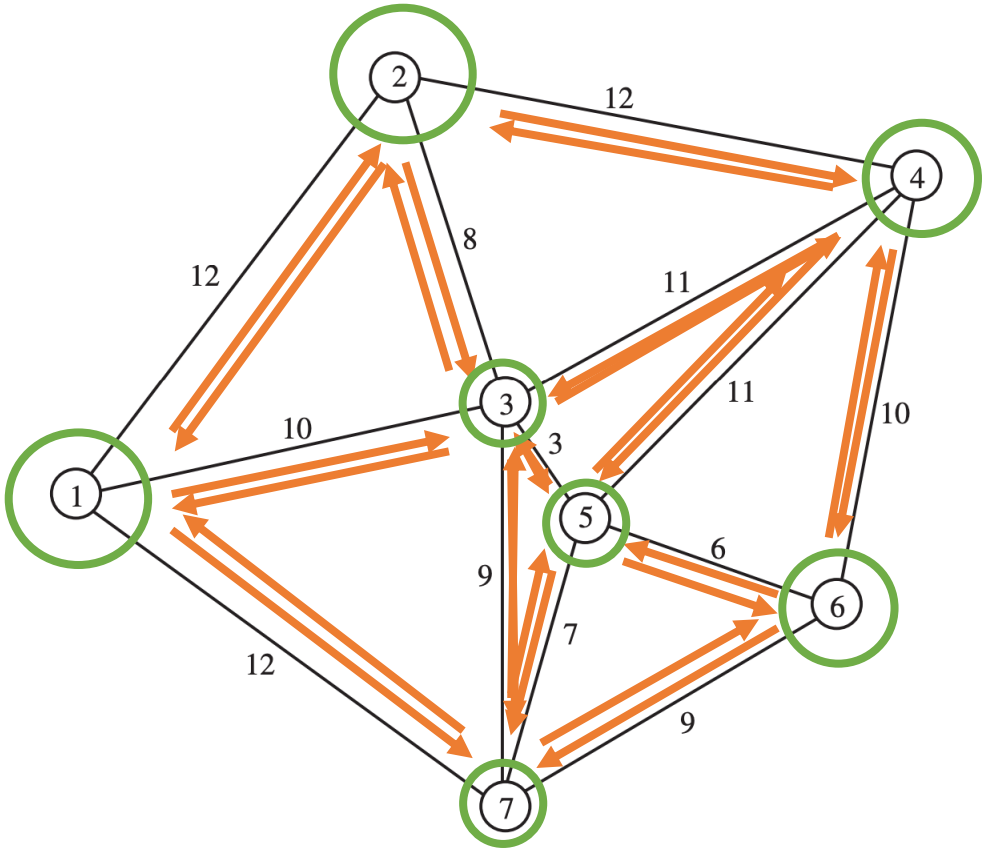


Local search algorithms and Optimisation Problems

Genetic Algorithms – Basic algorithm – TSP example

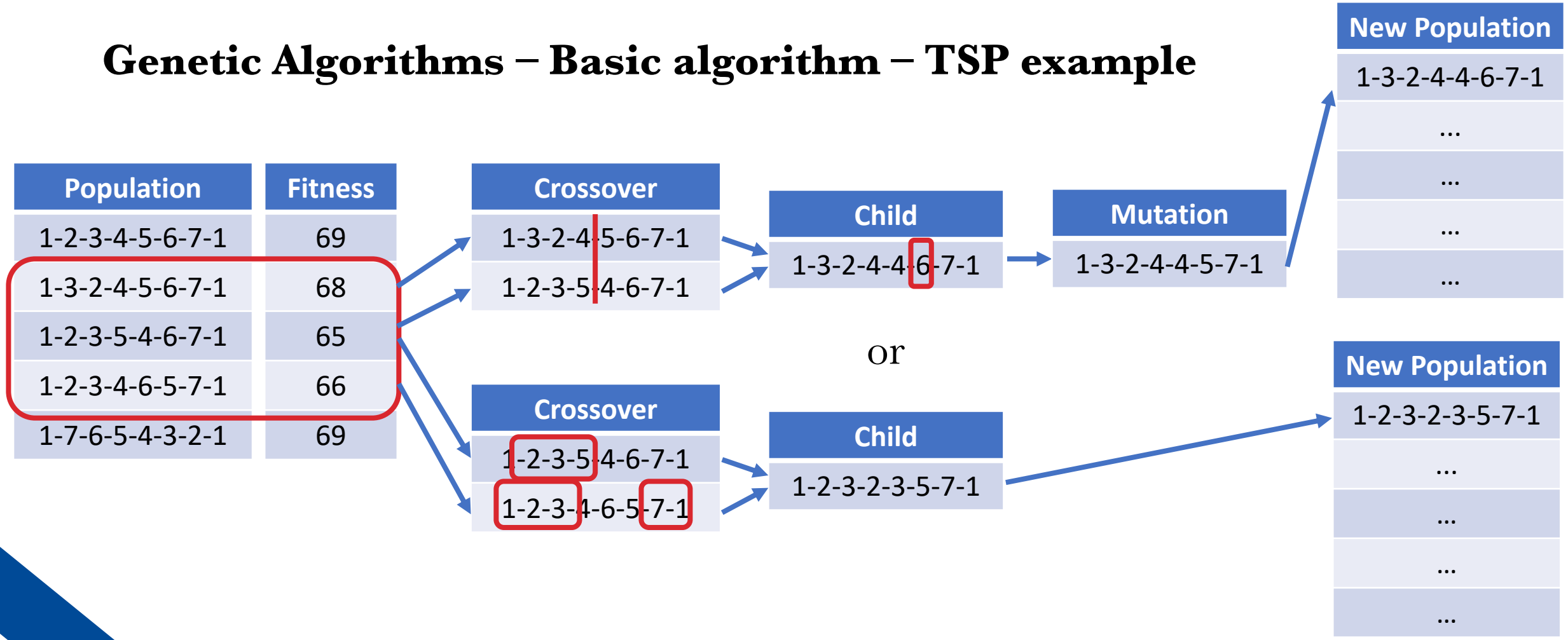
- 1 trial example

city	Options	Random Selection	Tour
1	1 – 2, 1 – 3, 1 – 7	1– 2	1 – 2
2	2 – 1, 2 – 3, 2 – 4	2 – 4	1 – 2 – 4
4	4 – 2, 4 – 3, 4 – 5, 4 – 6	4 – 6	1 – 2 – 4 – 6
6	6 – 4, 6 – 5, 6 – 7	6 – 5	1 – 2 – 4 – 6 – 5
5	5 – 6, 5 – 4, 5 – 3, 5 – 7	5 – 7	1 – 2 – 4 – 6 – 5 – 7
7	7 – 5, 7 – 6, 7 – 3, 7 – 1	7 – 3	1 – 2 – 4 – 6 – 5 – 7 – 3
3	3 – 7, 3 – 5, 3 – 4, 3 – 2, 3 – 1	3 – 1	1 – 2 – 4 – 6 – 5 – 7 – 3 – 1



Local search algorithms and Optimisation Problems

Genetic Algorithms – Basic algorithm – TSP example



Local search algorithms and Optimisation Problems

Genetic Algorithms – Basic algorithm – TSP example

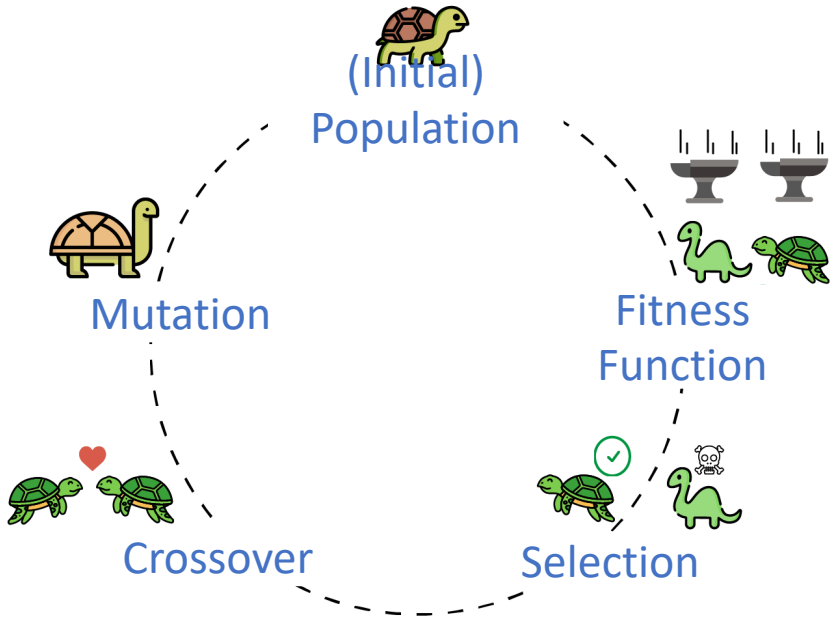
Member	Initial Population	Distance
1	1-2-4-6-5-3-7-1	64
2	1-2-3-5-4-6-7-1	65
3	1-7-5-6-4-2-3-1	65
4	1-2-4-6-5-3-7-1	64
5	1-3-7-5-6-4-2-1	66
6	1-2-4-6-5-3-7-1	64
7	1-7-6-4-5-3-2-1	65
8	1-3-7-6-5-4-2-1	69
9	1-7-6-4-5-3-2-1	65
10	1-2-4-6-5-3-7-1	64

Member	Parents	Children	Member	Distance
1	1-2-4-6-5-3-7-1	1-2-4-5-6-7-3-1	11	69
7	1-7-6-4-5-3-2-1	1-2-4-6-5-3-7-1	12	64
2	1-2-3-5-4-6-7-1	1-2-4-5-6-7-3-1	13	69
6	1-2-4-6-5-3-7-1	1-7-6-4-5-3-2-1	14	65
4	1-2-4-6-5-3-7-1	1-2-4-6-5-3-7-1	15	64
5	1-3-7-5-6-4-2-1	1-3-7-5-6-4-2-1	16	66

Local search algorithms and Optimisation Problems

Genetic Algorithms – Basic algorithm – TSP example

- When to stop?
  - Due to time
  - When a number of iterations are reached
  - When the best solution is not improving
  - When we reach a "good solution value"



https://www.generativedesign.org/02-deeper-dive/02-04_genetic-algorithms/02-04-01_what-is-a-genetic-algorithm

Local search algorithms and Optimisation Problems

Genetic Algorithms – Basic algorithm – TSP example

```
function GENETIC-ALGORITHM(population, fitness) returns an individual
  repeat
    weights ← WEIGHTED-BY(population, fitness)
    population2 ← empty list
    for i = 1 to SIZE(population) do
      parent1, parent2 ← WEIGHTED-RANDOM-CHOICES(population, weights, 2)
      child ← REPRODUCE(parent1, parent2)
      if (small random probability) then child ← MUTATE(child)
      add child to population2
    population ← population2
  until some individual is fit enough, or enough time has elapsed
  return the best individual in population, according to fitness

function REPRODUCE(parent1, parent2) returns an individual
  n ← LENGTH(parent1)
  c ← random number from 1 to n
  return APPEND(SUBSTRING(parent1, 1, c), SUBSTRING(parent2, c + 1, n))
```

A genetic algorithm. Within the function, *population* is an ordered list of individuals, *weights* is a list of corresponding fitness values for each individual, and *fitness* is a function to compute these values.

Local search algorithms and Optimisation Problems

Genetic Algorithms

Limitations/drawbacks:

- They are not effective in solving simple problems
- Lack of proper implementation may make the algorithm converge to a solution that is not optimal
- The quality of the final solution is not guaranteed
- Repetitive calculation of fitness values may make some problems to experience computational challenges