Lab 6 - Design and implement evolutionary algorithms to solve the travelling salesman problem

CSE, SUSTech

Outline of This Lab

- Teaching Assistant
- Travelling Salesman Problem
- How Does Crossover/Mutation Work
- Does The Search Operator Matter?
- Experimental Setup
- Illustrate The Results!

Teaching assistant

- Ms. Honglin Jin (12531321@mail.sustech.edu.cn)

Travelling Salesman Problem (TSP)

- 旅行推销员问题/最短回路问题
- Aim: find a shortest tour of *n* cities by visiting each city exactly once and returning to the starting city.

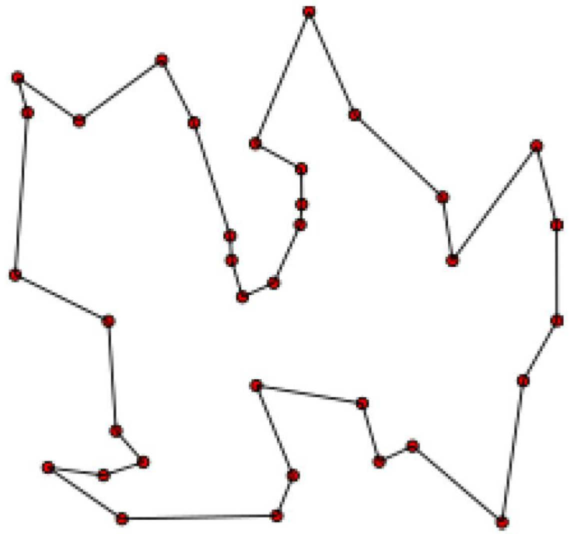


Figure 1: A solution to a TSP problem (figure from Wikipedia).

Formalisation of TSP

Given

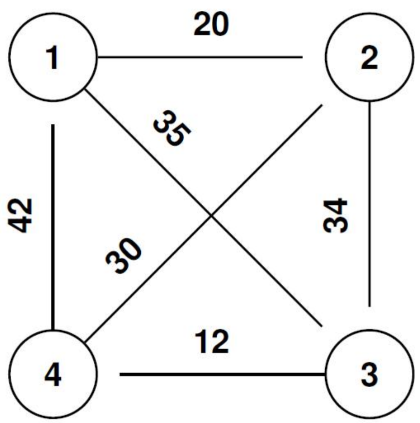
- ✓ n cities and
- ✓ $d_{i,j}$, the cost/distance of city i to city j, $\forall i, j \in \{1, . . . , n\}$,

among them, find a permutation ($x_1, x_2, . . . , x_n$) of ($1, 2, . . . , n$) such that

$$D = \sum_{i=1}^{n} d_{x_i, x_{i+1}}$$

is minimised, where $x_{n+1} = x_1$.

Distance Matrix
-*Example*

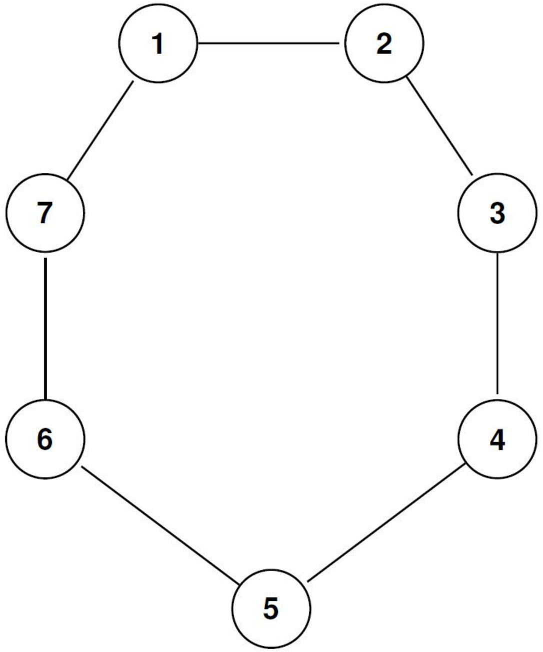


	1	2	3	4
1	-	20	35	42
2	20	-	34	30
3	35	34	-	12
4	42	30	12	-

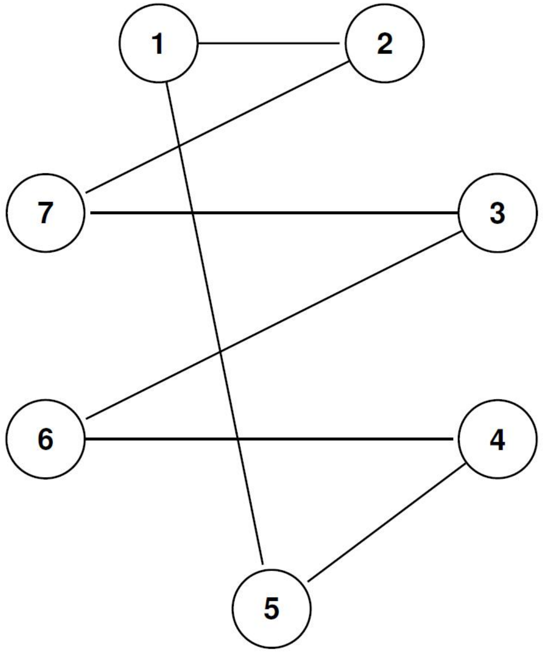
Example: $d_{1,2} = 20$**, the cost/distance of city** 1 **to city** 2**.**

Tour Representation
-Example

- $(x_1, x_2, \ldots, x_n)$ means that starting from city x_1, visiting x_2, then x_3, then . . . , then x_n and finally going back to x_1.



Example 1: $(x_1, x_2, x_3, x_4, x_5, x_6, x_7)$



Example 2: $(x_1, x_2, x_7, x_3, x_6, x_4, x_5)$

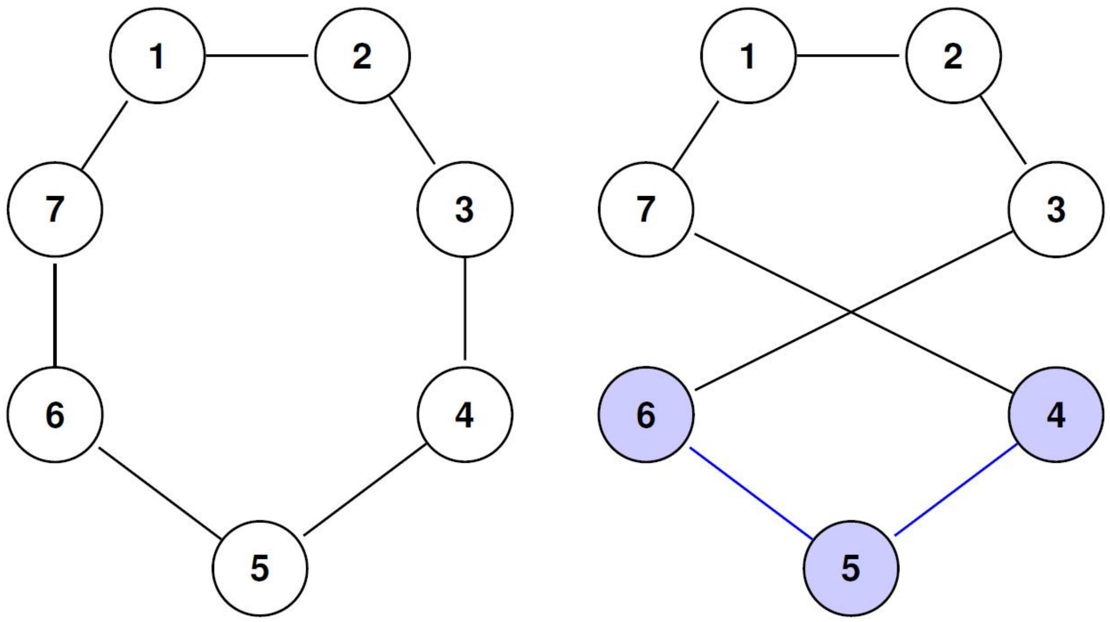
How Does Crossover/Mutation Work
-Simple Recombination (Crossover)

1. Construct an edge map from 2 parental tours.
2. Construct a child tour from the edge map.
   a) Choose the initial city at random as the current city.
   b) Determine which of the cities in the edge list of the current city has the fewest entries in its own edge list. The city with the fewest entries becomes the current city. Ties are broken randomly.

Please refer to lecture notes for more details

How Does Crossover/Mutation Work
- Reverse Mutation

Reverse any segment of the tour. Example:



$$(x_1, x_2, x_3, x_4, x_5, x_6, x_7) \to (x_1, x_2, x_3, x_6, x_5, x_4, x_7)$$

Tournament Selection

- At generation *t*, the population is G_t, of *P* solutions (tours)
  1. Generate *P* solutions by search operator(s).
  2. Each of the *2P* solutions (parents+offspring) competes against *α%* of the *2P* which are chosen at random. Note that:
     - ✓ A solution does not compete against itself.
     - ✓ The probability of s_i obtaining a win over the opponent s_o is

$$p_{s_i} = \frac{tourLength(s_o)}{tourLength(s_i) + tourLength(s_o)}.$$

     - ✓ Solutions with shorter tours have higher probabilities of winning.
     - ✓ *α* is a control parameter.
  3. The population G_{t+1} of next generation consist of the *P* solutions with most wins in the competition.

Does The Search Operator Matter?

- Implement the following three evolutionary algorithms
  - ✓EA1: simple recombination operator + tournament selection (*α = 10*) but no mutation.
  - ✓EA2: reverse mutation + tournament selection (*α = 10*) but no recombination.
  - ✓EA3: simple recombination operator + reverse mutation + tournament selection (*α = 10*).

Experimental Setup

- Five randomly generated TSPs with 20, 40, 60, 80, and 100 cities, respectively.
- Population size P = 10.
- Maximum generation numbers: 200

Illustrate The Results!

For each TSP instance, provide average results of running three EAs for 200 generations by using a table.