

Metaheuristic Optimization

Lab 1: TSP solution generator

Traveling Salesman Problem

The Traveling Salesman Problem (TSP) asks the following question: “Given a list of cities and the distances between each pair of cities, what is the shortest possible route that visits each city and returns to the origin city?”. It is an NP-hard problem with important practical and theoretical questions in computer science.

The input is a complete graph with weighted edges. For brevity each vertex $v \in V$ is given an x -coordinate $x(v)$ and y -coordinate $y(v)$. The weight of the edge between vertex u and vertex v is defined by,

$$w(u, v) = \text{nint} (\text{sqrt}((x(u) - x(v))^2 + (y(u) - y(v))^2))$$

Where *nint* is a function that rounds a number to the nearest integer. The objective is to find the shortest cycle in the graph where each node is visited exactly once.

TSP has obvious applications in route planning and other logistics scenarios, but also in wire routing for printed circuit boards, genome analysis, and many other domains. Because it is easy to explain but computationally challenging, it is often used as a testbed for new algorithms to solve combinatorial problems.

In this lab you need to implement the following heuristic solution for the TSP:

Nearest neighbor insertion : starts with randomly selected city and insert each new city into the current tour after the city to which it is closest. (If there is more than one city to which it is closet, insert it after the first such city you encounter).

I/O Specification

Format: the input file contains $n+1$ lines, where n is the number of nodes in the graph, i.e., $|V|$. The first line contains an integer n , it is followed by n lines, each line corresponds to the x, y -coordinates of a node. The coordinates contain integer and decimal values

[input format]

n

$ld_0 \ x_0 \ y_0$

$ld_1 \ x_1 \ y_1$

...

$ld_{n-1} \ x_{n-1} \ y_{n-1}$

The output file contains $n+1$ lines. The first line has an integer L with the length of the cycle (notice that all edge weights are integral). The $n+1$ following lines define the order that the nodes are visited in the cycle, with one node per line.

[Output format]

L

ld_1

ld_2

ld_3

...

ld_n

Example

[Input Example]

4

1 823170 415922

2 793699 274913

3 981665 218777

4 878910 431320

[Output Example]

2180104.3

1

2

3

4

Lab work

Write a python program to solve a given TSP instance (implementing the Nearest Neighbor Insertion heuristic).

You need to run your code using the command: `python tsp.py <input filename> <output filename>`

For instance: “`python tsp.py instance.tsp sol.tsp`” means the problem will use “`instance.tsp`” and creates a new file `sol.tsp` with your solution.

Note: A test dataset (TSPdata.zip) is available on Canvas.