

The following heuristics are used by the program to generate a tour for the Travelling Salesman Problem:

1. Nearest Neighbour Heuristic

TSP : Greedy Constructive Methods

Nearest Neighbour Heuristic
Start at some city.
Move to nearest neighbour
as long as it does not close the loop prematurely

An example run →

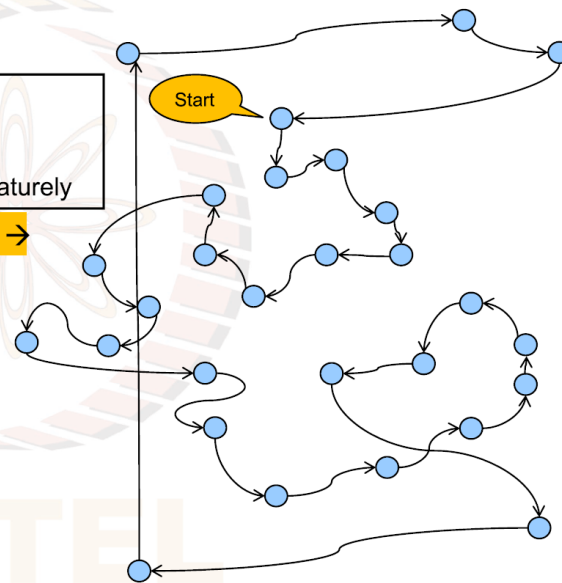


Figure 1: Nearest Neighbour Heuristic

2. Variation of Nearest Neighbour Heuristic

Variations:

Extend the partial tour at either end.

Figure 2: Nearest Neighbour Heuristic on both sides

3. Greedy Algorithm

TSP : Greedy Constructive Methods

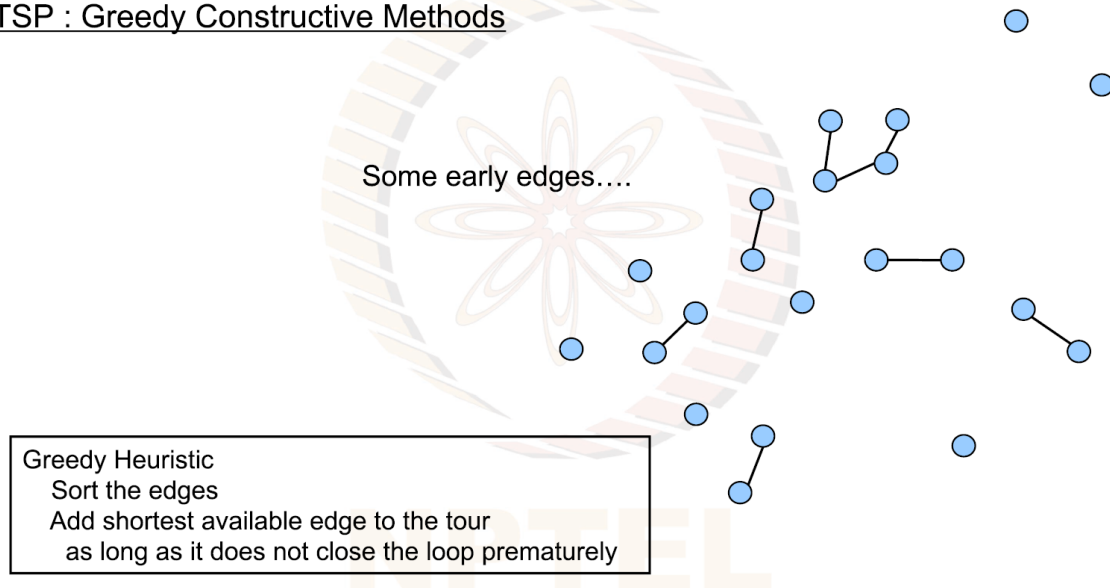
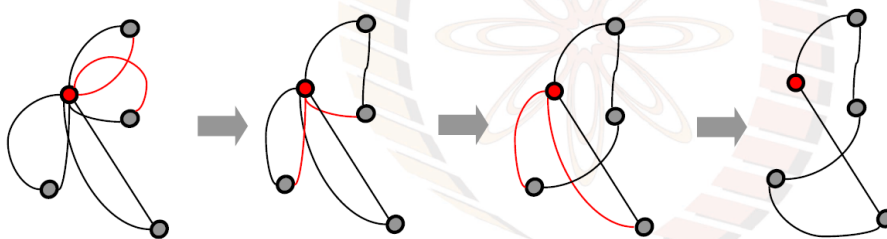


Figure 3: Greedy Algorithm

4. Saving Heuristic

Savings Heuristic

The Savings Heuristic starts with $(n-1)$ tours of length 2 anchored on a base vertex and performs $(n-2)$ merge operations to construct the tour.



Remove two edges connected the base vertex from two loops and add an edge to connect the two hanging vertices.

The two edges are chosen such that there is maximal savings of (total) cost in the resulting graph.

Figure 4: Saving Heuristic

Variation of Nearest Neighbour heuristic where edges can be added from both sides seems to be perform best for the given examples(euclidean100.txt,non euclidean100.txt).

Initially, I started off with the Nearest Neighbour heuristic and extended to allow edges to be added from either sides. In both of these algorithms, I consider all the n vertices as starting vertex and create a tour and choose the best out of these.

Then, I went ahead implementing the Greedy heuristic but it didn't improve on the tours generated by the nearest neighbour heuristic.

Finally, I tried implementing the Saving Heuristic and ran into small implementation issues and wasn't able to complete the implementation.