Solving the Travelling Salesman Problem (TSP) by local search

The motivation behind the Travelling Salesman Problem (also known as Travelling Salesperson Problem) is the problem faced by a salesperson who needs to visit a number of customers located in different cities and tries to find the shortest round trip accomplishing this task. In a more general and abstract formulation, the TSP is, given a directed, edge-weighted graph, to find a shortest cyclic path that visits every node in this graph exactly once. Often, the TSP is defined in such a way that the underlying graphs are always complete graphs, that is, any pair of vertices is connected by an edge.

In this assignment task, we assume that TSP instances are specified as complete graphs. In addition, we assume the vertices correspond to the points in a Euclidean space, and the edge weight between any two vertices corresponds to the Euclidean distance between the respective points.

In local search paradigm, we start at some position in search space and iteratively move from the current position to a neighbouring position. In constructive search, the search space contains partial candidate solutions and the search step consists of doing extension with one or more solution components. In perturbative search, the search space contains complete candidate solutions and the search step consists of doing modification of one or more solution components.

Both the constructive and the perturbative search can be described under the local search paradigm. While in the literature, the term local search is mostly used for perturbative search methods, it also applies to constructive search methods. Perturbative search methods start from some complete candidate solution. The task of generating such candidate solutions is commonly accomplished by constructive search methods or construction heuristics. Constructive search methods iteratively extend an initially empty candidate solution by one or several solution components until a complete candidate solution is obtained. Constructive search methods can thus be seen as operating in a search space of partial candidate solutions.

Constructive search methods

The *Nearest Neighbour Heuristic (NNH)* starts tour construction from some randomly chosen vertex$u1$ in the given graph and then iteratively extends the current partial tour $p = (u_1, . . ., u_k)$ with an unvisited vertex u_{k+1} that is connected to u_k by a minimum weight edge (u_{k+1} is called a *nearest neighbour* of u_k); when all vertices have been visited, a complete tour is obtained by extending p with the initial vertex, u_1. The tours constructed by the NNH are called *nearest neighbour tours*.

Insertion heuristics construct tours in a way that is different from that underlying the NNH; in each step, they extend the current partial tour p by inserting a heuristically chosen vertex at a position that typically leads to a minimal length increase. Several variants of these heuristics exist, including

(i) *Nearest insertion* construction heuristics, where the next vertex to be inserted is a vertex u_i with minimum distance to any vertex u_j in p;

(ii) C*heapest insertion*, which inserts a vertex that leads to the minimum increase of the weight of p over all vertices not yet in p;

(iii) *Farthest insertion*, where the next vertex to be inserted is a vertex u_i for which the minimum distance to a vertex in p is maximal;

(iv) R*andom insertion*, where the next vertex to be inserted is chosen randomly.

The ***Greedy** Heuristic*: The construction heuristics discussed so far build a complete tour by iteratively extending a connected partial tour. An alternative approach is to iteratively build several tour fragments that are ultimately patched together into a complete tour. One example for a construction heuristic of this type is the *Greedy Heuristic*, which works as follows. First, all edges in the given graph G are sorted according to increasing weight. Then, this list is scanned, starting from the minimum weight edge, in linear order. An edge e is added to the current partial candidate solution p if inserting it into G', the graph that contains all vertices of G and the edges in p, does not result in any vertices of degree greater than two or any cycles of length less than n edges.

For construction heuristics, you may also consider the simple Minimum Spanning Tree (MST) based heuristic and the Christofides heuristic.

Perturbative search methods

2-opt and ***3-opt*** are iterative improvement algorithms based on k-exchange neighbourhoods. Various speed-up techniques, including fixed-radius near neighbour search and candidate lists may be considered. ***Or-opt*** and ***2.5-opt*** are closely related to 2-opt and 3-opt algorithms.

In addition, ***Node Shift*** and ***Node Swap*** are simple iterative improvement algorithms. A Node Shift neighborhood is generated by shifting a city to other position in tour, while a node swap neighborhood is generated by swapping two cities in tour.

In this assignment, you need to implement at least three heuristics from each of the constructive and the perturbative category and present the result of the combinations. You should run experiments on the provided 21 benchmark data sets. For experimentation, you may consider semi-greedy versions in the constructive category. For example, instead of the nearest candidate, you may randomly select any of the top 3 or top 5 vertices in Nearest Neighbour Heuristic (NNH). Similar semi-greedy versions may be considered for nearest/cheapest/farthest insertion heuristic where you may select any of the top 3 or top 5 of the nearest/cheapest/farthest candidates for insertion for extending the current partial solution.