Report DO Assignment 3 - TSP

Context

The travelling 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 exactly once and returns to the origin city? It is an NP-hard problem in combinatorial optimization, important in operations research and theoretical computer science.

A brute force approach is not feasible even for moderately large data sets (O(n!)). In this assignment we explore some approximation algorithms for getting a good answer in a reasonable amount of time.

Results

Data set	Number of cities	Shortest path length
1	51	433.086190637
2	100	21547.4158296
3	200	30624.4645968
4	574	38623.9145733
5	1889	332307.903443

Algorithms:

Nearest Neighbor

Also known as the greedy solution, this algorithm consists of defining a starting point then always heading to the closest available point to build the route. It rapidly gives an initial solution with which we can then work on optimizing.

The algorithm requires finding the closest point among those available, meaning that the run time is O(n^2).

N.B Starting from a different point changes the solution, giving us a variety of different solutions to start from

Two Opt

The basic principle of Two Opt is to swap 2 different edges in the hope of finding a shorter overall distance. The main idea is to take a route that crosses over itself and rearrange it so it does not. We can do this according to the following pseudo code (source Wikipedia):

```
2optSwap(route, i, k) {
       1. take route[1] to route[i-1] and add them in order to new_route
       2. take route[i] to route[k] and add them in reverse order to
new_route
```

```
        3. take route[k+1] to end and add them in order to new_route
        return new_route;
    }
```

2-opt is a local search technique. This means we start at one possible solution, and we try to move to neighboring possible solutions.

Remove Intersections

This idea goes hand in hand with 2-opt. To efficiently apply 2-opt, we first need to detect where the path crosses over itself. To do this we use the Bentley Ottman algorithm. This is a line sweep algorithm and, for an input consisting of n line segments with k crossings, takes time O((n + k) log n).

This gives us the list of segment which intersect, on which we then apply 2-opt to get a solution with no intersections.

Simulated annealing

This is a probabilistic technique which we use in combination with two opt search. With local search, we can sometimes get stuck in local minima (by always searching for improvement), and this method aims to get us out of them. To do so the idea is to sometimes accept a less then optimal solution in the aim of taking us into a new neighborhood from which we will be able to find the optimal solution.

We go through all the combinations of 2-opt search, and on occasion we accept a sub-optimal solution with the following probability: exp(-delta/t), where delta is the difference in difference between the current best solution and the new proposed solution, and t is a temperature variable. Thus the closer solutions are to the current best solution, the more likely we are to choose them.

The unique idea behind simulated annealing is to vary the temperature during the search, starting with a high temperature (more randomness) and lowering it bit by bit (local search).

A couple of other optimisations are used as well:

A first idea is that long edges are more likely to be bad, so we start our 2-opt by trying to swap long edges for shorter edges. To do so we start by identifying edges in the route which are long i.e. we sorted the segments by length. We then start 2 opt by trying to swap the long edges with points that are closer to the original point.

A second idea is to reheat if after having tried all possible pairs and no better solution has been found. This could help us escape a local minina.

We terminate the algorithm if no improvement have been found for the last x seconds, with x=30 for example.

- Sim_anneal1: Tries to swap long edges with shorter edges during two opt loop. To do so, sorts edges by length and goes through them in reverse order. We also calculate the distance matrix between all points to find the points which are closest to the current point
- Sim_anneal2: Select edges randomly during 2-opt loop