CMSC 818B: Decision-Making for Robotics (Fall 2019)

Homework 1

Due: September 16th, 12PM (noon)

September 5, 2019

You must work on this homework individually. Your submission must be your original work. Please follow the submission instructions posted on canvas exactly.

Problem 1 — 70 points

Implement the Minimum Spanning Tree (MST) based 2–approximation algorithm for solving the metric Traveling Salesperson Problem (TSP). As discussed in class, once you generate the MST tour, you can further improve the tour length using a number of heuristic. Implement at least one such heuristic of your choice.

The input to your implementation will be a `.tsp` file that uses the TSPLIB format.[1] There are many ways of specifying the input graphs in the TSPLIB format. We will only use the `EDGE_WEIGHT_TYPE : EUC_2D` format where the input file will contain the coordinates of the vertices in the graph. The edge weights are not directly given in the graph but can be calculated by computing the Euclidean distance between the coordinates of two vertices. A sample input file is provided. The test input files will be similar and will only include the keywords shown in the sample: (`NAME, COMMENT, TYPE, DIMENSION, EDGE_WEIGHT_TYPE, NODE_COORD_SECTION, EOF`. The documentation for the keywords is given online.[2]

The output of your implementation should be a file named `<filename>.out.tour` if the input file is named `<filename>.tsp`. A sample input/output file is provided. The output file should contain the following keywords:

- `NAME : <filename.out.tour>`
- `COMMENT : Tour for <filename>.tsp (Length <length of the tour found using your implementation>)`
- `TYPE : TOUR`
- `DIMENSION : <number of vertices in the graph, also found in the input file>`
- `TOUR_SECTION`

The tour section contain a list of node/vertex indices in the order they appear in your output tour, one per line, followed by -1 and `EOF`. Do not include the coordinates of the nodes/vertices, only their indices. Refer to the sample output file.

You may use C/C++/Python/MATLAB to implement your algorithm. You must implement your algorithm from scratch and cannot use any online library. It is okay to use libraries that implement graph data structures; however, any code for finding the MST or the tour must be your own. If you are in doubt, check with the instructor.

[1] `https://wwwproxy.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/`

[2] `https://wwwproxy.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/tsp95.pdf`

Submission Instructions. You must submit a pdf report that gives a short description of your implementation. In particular, explain how you compute the MST and what heuristics you use for improving the solution (if any). Run your algorithm with the `eil51, eil76, eil101` input instances given on ELMS. In a table, report the length of the final tour found using your implementation, the optimal tour lengths for these instances that can be found in the TSPLIB documentation, the length of the MST, the length of the tour without any improvement heuristics, and the time taken (in seconds) to solve this instance.

Create at least 10 random instances with 100, 200, and 300 vertices each. In the pdf report, include a separate table (without the optimal tour length column) similar to the one mentioned above for these random instances.

Submit your neatly commented code, all the input/output files for the three aforementioned instances (not the random ones), as well as one README file that describes exactly how your code is to be run. In particular, list any external dependencies and how the input filename is given to your implementation.

Submit a screencast of you running your code on your machine. You can upload your screencast video on YouTube or Box or Google Drive and include a publicly-accessible URL in the pdf report.

Problem 2 — 30 points

Design an algorithm for solving the k robot metric TSP. The input is a complete metric graph, a starting vertex s, as well as an integer $k > 1$. The output should be k tours that all start and return to s such that each vertex in the input graph is on at least one tour and the length of the longest tour is minimized.

In the pdf report, write down the pseudocode for your algorithm. Explain the pseudocode and your rationale behind the algorithm. Analyze the performance of your algorithm. Your analysis could be theoretical (showing approximation bound or showing sample instances where the algorithm works poorly) or empirical (implement the algorithm and test it with random instances).