CPTS 580 Project Report

A heuristic algorithm for Traveling Salesman Problem

Yang Zhang
11529139

***Abstract*—This report focus on solving Traveling Salesman Problem (TSP) using heuristic algorithm called double the tree (the algorithm 6.4.2 in *the Applied Graph Theory by Gross and Yellen*) The first part of this report is to explain traveling salesman problem briefly. Then I will talk about the implementation detail and results.**

I. Introduction

The traveling salesman problem (TSP) is originally from a practical question "*Given a list of cities and the distances between each pair of cities, what is the shortest possible rout that visits each city exactly once and returns to the origin city?*" To talk this in computer way is to find the Hamiltonian Cycle with minimum cost for given sample connected graphs. TSP is not hard to understand, however, to find the optimum solution of TSP is an NP-hard problem in combinatorial optimization.

The problem was first formulated in 1930 and is one of the most intensively studied problems in optimization. It is used as a benchmark for many optimization methods. Even though the problem is computationally difficult, a large number of heuristics and exact algorithms are known, so that some instances with tens of thousands of cities can be solved completely and even problems with millions of cities can be approximated within a small fraction of 1%. [1]

II. The Algorithm

In this project, the Algorithm 6.4.2 *Double the Tree* is applied and implemented. The difference between 6.4.2 and other algorithms such as Algorithm 6.4.1 *Nearest Neighbors* is that Algorithm 6.4.2 guarantees performance for graphs that satisfying the triangle inequality.

The implementation of Double the Tree Algorithm follows the guild line below [2],

Input: A weighted complete graph G.

Output: A sequence of vertices and edges that forms a Hamiltonian cycle.

1. *Find a mst T* of G.*
2. *Create an eulerian graph H by using two copies of each edge of T*.*
3. *Construct an eulerian tour W of H.*
4. *Construct a Hamiltonian cycle in G from W as follows:*

   Follow the sequence of edges and vertices of W until the next edge in the sequence is joined to an already visited vertex. At that point, skip to the next unvisited vertex by taking a shortcut, using an edge that is not part of W. Resume the traversal of W, taking shortcuts whenever necessary, until all the vertices have been visited. Complete the cycle by returning to the starting vertex via the edge joining it to the last vertex.

Implementation notes:

Input is the adjacent matrix of the graph.

Output is sequence of resulting Hamiltonian tour.

Throw an exception if the size of input graph either smaller than 5 or bigger than 20.

Throw an exception if the input graph is not complete.

For finding the MST of the input graph, I used the implementation of Prim's Algorithm from hw5.

For finding shortcuts of the eulerian tour, using a array to record the current tour, for any other vertex

V, if its parent is part of the tour and V is not in the our yet, adding V to the tour.

III. Results

Running the algorithm over the graphs from exercises 6.4.2 – 6.4.5 in the book, I got the following results:

```
Exercise 6.4.2
a -> d -> e -> c -> b -> a
The total cost: 15

Exercise 6.4.3
a -> d -> c -> e -> b -> a
The total cost: 29

Exercise 6.4.4
       a -> c -> e -> d -> b -> f -> a
The total cost: 36

Exercise 6.4.5
a -> f -> e -> b -> c -> d -> a
The total cost: 35
```

For exercises 6.4.2 and 6.4.3, I find the optimal cost of a Hamiltonian cycle by exhaustively trying out all possibilities, which are 15 for 6.4.2 and 29 for 6.4.3. The results I found by hand are the same as the solutions from the program. Moreover, those results are obviously within the multiplicative factor of 2 from the optimal cost. This also follows the theorem 6.4.4 in the book, because both graph 6.4.2 and graph 6.4.3 obey the triangle inequality.

References

[1] World TSP
http://www.math.uwaterloo.ca/tsp/world/

[2] Graph Theory and Its Application by Gross

Page 277