Traveling Salesman Experiment

Team JS: Jonathan Irvin Gunawan, Khor Shi-Jie
October 25, 2014

1 Introduction

The study of traveling salesman problem (TSP) is a natural generalization of the Hamiltonian cycle problem. We are interested to identify a cycle within a weighted graph such that the cycle includes all vertices in the graph and each vertex is only visited once in the cycle. In addition, we would to identify the Hamiltonian cycle with the minimum sum of edge weight. More formally, A close analogue to this problem is the Chinese postman problem which seeks to identify a tour of minimum weight that traverses each edge in the graph at least once. While the exact solution to the Chinese postman problem can be obtained in polynomial time, it is surprising that a subtle change in the requirement of the problem makes the Travelling Salesman Problem unsolvable in polynomial time. In particular, we note that

Theorem 1. *TSP is NP-Hard.*

Proof. We will reduce TSP into a directed Hamiltonian Cycle Problem, which is known to be a NP-Hard problem. Given a directed graph G, we construct a weighted complete directed graph G' which has the same vertices as G. In addition, we assign a weight of 1 to the edges in G which were originally in G', and a weight of ∞ otherwise. Suppose we found a solution to the TSP using graph G' that has finite weight. Then, each of the edges must have weight 1 and hence must be present in the original graph G. As such, the solution to the TSP in G' is also a solution to the Hamiltonian Cycle Problem in G. On the other hand, if the solution to TSP in G' has infinite weight, then there is no Hamiltonian cycle which only consist of edge weight 1. Hence, there is no Hamiltonian cycle in G. As such, the directed Hamiltonian Cycle Problem reduces to TSP, which proves that TSP is NP-Hard. □

Due to the impossibility of find a polynomial-time algorithm to solve TSP and the relevance of TSP to real life problems, there has been a lot of research that seeks to find an efficient approximation algorithm with a small approximation ratio. Within our course, we have explored several approximation algorithm to solve the TSP, including a 2-approximation algorithm based on constructing a minimum spanning tree (MST), and a 1.5-approximation algorithm that include the application of a minimum weight perfect matching algorithm. However, the effectiveness of these algorithms are only theoretical in nature. While the worst-case bounds for these algorithms are irrefutable, the optimality of these algorithm may differ from our expectations when we apply them to real world data (i.e. a 2-approximation algorithm may perform better on average than a 1.5-approximation algorithm). As such, this project aims to compare the suitability of various approximation algorithm when applied to graphs obtained from real world data or other random means.

2 Problem Formulation

Generally, there are four variants of TSP that one can seek to solve. We can either have a graph in which the distance function is a metric, or a graph with a generic distance function. In addition, we can also choose to allow repeated vertices within our solution. The four variants are summarized in the table below:

	Repeats	No Repeats
Metric	M-R TSP	M-NR TSP
General	G-R TSP	G-NR TSP

We have shown in class that M-R TSP, M-NR TSP and G-R TSP are equivalent. For our project, we are more interested in M-NR TSP as we are using data obtained from the real world. In addition, we will assume that the graph is non-directional, and the distance between any two vertices is well-defined.

For this project, we will implement four different approximation algorithms and profile the performances on a set of random graphs and real world data. The four approximation algorithms are:

1. *Nearest Neighbour Heuristics.* We start from a cycle with only one vertex in the cycle. Then, while there are vertices which are not included within our cycle, we choose a vertex which is closest our cycle. Then, we add the vertex to the cycle at a position whereby the increase in the length of the cycle is minimal. Once all the vertices are added to the cycle, we will obtain a $O(log n)$-approximate solution to the TSP (to be proven in the next section).

2. *Minimum Spanning Tree.* We will build a minimum spanning tree for the graph and conduct a DFS traversal on the graph. Within the DFS traversal, we will skip any vertices which has already been visited. This will produce a 2-approximate solution to the TSP (proven in class).

3.

4.

3 Algorithms and Theory

4 Implementations and Experiments

5 Conclusions

6 Bibliography

References

[1] J. Clark and D. Holton. *A First Look at Graph Theory.* World Scientific, 2005.