Traveling Salesman Problem Proposal

Proposed by Anqi Wang, Rakesh Gururaj

1. **Problem Statement**

Traveling Salesman Problem is that given a set of cities and the distance between every pair of cities, the problem is to find the shortest possible route that visits every city exactly once and returns to the starting point. To summarize, the problem is to find the shortest path go through each vertex once and then return back to start vertex in the graph and It can be seen as a minimum weight Hamiltonian Cycle problem.

We are focused on solving the problem using 4 different approaches: Brute force, Dynamic Programming, Greedy, and Minimum Spanning Tree(MST). Later, we shall be analyzing the performance metric in terms of running time and space involved in the above-mentioned approaches.

There are many applications of this problem such as the cheapest airplane tours. This will help people save money and time when traveling around the country. There are also many derived problems. For example, the traveling baseball fan problem. It is a harder version of TSP because it will consider the baseball scheduling constraints. And another application is to schedule the collection of coins from payphones throughout a given region. So it is important and useful to solve TSP and apply it in the real world.

Our work is to solve the TSP problems with serval approaches and compare the performance of those solutions. There are different sizes of the data set and we will evaluate the performance based on these data set and find the best solution.

1. **Programming language**

Java, We shall leverage hprof for profiling the CPU performance which will later be used for analysis.

1. **Proposed approaches**

Mathematical formulation:

Let G=(V, E) be the complete graph (or digraph) with n nodes and let w be the length of e ∈ E, Let H be the set of all hamiltonian cycles tours in K Find min{ w(T) | T ∈ H}.

Approach 1: Brute Force:

The most time-consuming and optimal approach to the problem is to choose the brute force technique. When there are n cities given to us, we shall try to find all the possible routes from the initial point and travel to n-1 cities and get back to the initial point. After exploring all the possible routes, we will be choosing the shortest route amongst all. However, this approach has a direct implication on the number of cities. As the number of cities grows, the problem becomes complex since with n cities, we need to explore (n-1)! routes. To illustrate the approach we shall use a minimum number of cities such as 4 cities and find the shortest path.

Approach 2: Dynamic programming:

Let vertex 1 be the start point and the endpoint. The minimum cycle is Minimum of all cost(i) + dist(i,1). cost(i) is a minimum cost path with 1 as the starting point, i as the ending point and all vertices appearing exactly once. dist(i, 1) is the distance from i to 1.

Here use DP to calculate the cost(i). The sub-problem is C(S, i) be the cost of the minimum cost path visiting each vertex in subset S exactly once, starting at 1 and ending at i. Starting the size of 2 and then increase the size.

Approach 3: Greedy - Dijkstra's shortest path:

This approach will focus on starting from a city and calculating the shortest distance between neighboring cities and choosing the nearest neighbor that has not been visited. This process is continued until we explore all the cities at least once. By making the local choices we try to find the shortest distance of the cycle.

Approach 4: Minimum Spanning Tree(MST):

This approach focuses on cutting down the number of edges between the cities. We first build a tree with cities as the nodes of the tree. We then cut down the edges and construct a minimum spanning tree connecting all the cities with the least number of edges. Then we shall select the starting point and construct a path using the MST.

1. **Data Set**

We will be taking advantage of the publicly available dataset for solving the problem.

The dataset is downloaded from the below website:

<https://people.sc.fsu.edu/~jburkardt/datasets/cities/cities.html>

We shall be using the below-mentioned dataset for analyzing the solution implemented using different algorithms.

4 cities:

CO04

GRID04

15 cities:

LAU15

30 international cities:

HA30

1. **References**

|  |  |
| --- | --- |
| **[1]** | http://co-at-work.zib.de/berlin2009/downloads/2009-09-21/2009-09-21-1600-MG-TSP-and-Applications.pdf |
| **[2]** | <https://www.geeksforgeeks.org/travelling-salesman-problem-set-1/> |
| **[3]** | <https://people.sc.fsu.edu/~jburkardt/datasets/cities/cities.html> |
| **[4]** | <https://docs.oracle.com/javase/8/docs/technotes/samples/hprof.html> |
| **[5]** | Bellman, Richard. "Dynamic programming treatment of the traveling salesman problem." Journal of the ACM (JACM) 9.1 (1962): 61-63. |
| **[6]** | Ejim, Samson. (2016). Implementation of Greedy Algorithm in Travel Salesman Problem. 10.13140/RG.2.2.23921.48485. |