**CSE 545 – Artificial Intelligence – Project 1 – TSP Problem**

Jordan Richardson

BA Computer Science

Speed School of Engineering

University of Louisville, USA

[Jkrich05@louisville.edu](mailto:Jkrich05@louisville.edu)

**Introduction**

This project focused on certain aspects of solving Travelling Salesperson Problems using the Brute Force method. The background of this type of problem is easy to understand but provides a challenge, nonetheless. You are a salesman given a list of cities that you must visit once, before you return home. The goal of this problem is to find the least possible distance or cost required to complete the trip, otherwise known as a Hamiltonian Cycle. This project was done with hopes we gain and hone the skills of generating all permutations of a given set of numbers, the ability to trace a Hamiltonian path through an undirected graph, and the ability to properly use brute force to calculate the minimum cost of our path.

**Approach**

The algorithm we are using for this problem is called Brute Force. This specific algorithm is not the most efficient method for solving a TSP problem but is straightforward. This algorithm involves calculating all possible permutations of your cities and calculating the total distance for each permutation. The shortest route among all the permutations possible would be the best path.

**Results**

The algorithm performs great for basic TSP problems, however as the number of nodes increase, efficiency exponentially explodes to a halt past certain sizes. In my solution, I noticed that each city past 9 nodes increase by a minimum of 10 times the previous evaluation timeframe. This means that by 12 nodes it was around 2000 seconds or 30 minutes, and 13 nodes would have likely taken greater than 5 hours. This basically means that the brute force method can only be used in conjunction with other methods to increase efficiency or on a very small scale by itself.

**Results – Data**

For this project we were provided with .TSP files that contained some basic information about how it was generated up to line 7, and a list of cities with longitude and latitude separated by whitespaces after line 7. Our first problem had 4 cities, increasing by one until we reached a max of 12 cities for this specific project.

**Results – Results**

**4 Cities** **5 Cities**

Elapsed Time: 0.002000093460083008 Elapsed Time: 0.0029993057250976562

Optimal Tour: [1, 4, 2, 3] Optimal Tour: [1, 2, 5, 3, 4]

A diagram of a problem

Description automatically generated A diagram of a problem

Description automatically generated

**6 Cities**   **7 Cities**

Elapsed Time: 0.0040035247802734375 Elapsed Time: 0.019316673278808594

Optimal Tour: [1, 2, 3, 4, 5, 6] Optimal Tour: [1, 2, 7, 3, 6, 5, 4]

A diagram of a problem

Description automatically generated A diagram of a problem

Description automatically generated

**8 Cities 9 Cities**

Elapsed Time: 0.12251734733581543 Elapsed Time: 1.2435343265533447

Optimal Tour: [4, 5, 2, 3, 7, 1, 6, 8] Optimal Tour: [6, 7, 1, 8, 4, 9, 2, 5, 3]

A diagram of a problem

Description automatically generated A diagram of a problem

Description automatically generated

**10 Cities 11 Cities**

Elapsed Time: 13.750495433807373 Elapsed Time: 157.00639510154724

Optimal Tour: [1, 2, 7, 6, 8, 5, 9, 10, 4, 3] Optimal Tour: [1, 6, 10, 11, 8, 9, 7, 5, 3, 4, 2]

A graph with red and blue lines

Description automatically generated A graph with blue lines and dots

Description automatically generated

**12­ Cities**

Elapsed Time: 2035.443779706955

Optimal Tour: [1, 8, 2, 3, 12, 4, 9, 5, 10, 6, 7, 11]

A graph with red arrows and blue lines

Description automatically generated

**Discussion**

During this assignment, I learned a lot about principles that weren’t covered very deeply in other classes. I started out building my assignment to give me as much information as possible as it progresses. However, I quickly realized how much this can cost efficiency wise when the problem with 10 nodes took over an hour too run, and when 12 nodes caused my shell to lose connection. I found that the combination of mathematics and programming made more sense to me than many other projects and highlighted the capabilities of computers. In general, my results were obtained almost instantaneously up until the 9th city, upon which the permutations are increasing drastically with each node added. The time of completion for each node after the 9th can be estimated somewhat as the timing seems to linearly increase with the permutation total. To generate all the permutations of my problems I used itertools, along with a combination of initializations and counters. It calculates all possible permutations of however many cities it reads in, and as it progresses through them, each time it finds a smaller value for total\_distance than the current best\_distance, it replaces that value and once it finishes progressing through all the permutations it prints the best path and total\_distance.

**References**

I did not use anything specifically to base my code off of, however I did read up more on the TSP and other approaches for it before starting to design my own.