**Travelling Salesperson Problem ‐ Brute Force**

Andrew Nguyen Vo

Computer Engineering and Computer Science

Speed School of Engineering

University of Louisville, USA

[awnguy01@louisville.edu](mailto:awnguy01@louisville.edu)

1. **Introduction** (What did you do in this project and why?)

The problem posed by this project was to find a programmatic answer to the Traveling Salesperson problem. Given a dataset of cities with xy-coordinates, the requirement was to find the Hamiltonian cycle visiting each city exactly once before returning to the starting city. The stipulation was to find this minimum cost route by generating each permutation using a brute force approach in order to illustrate the limitations of this approach on increasingly larger datasets.

1. **Approach** (Describe algorithm you are using for this project)

*Solution produced using Python 3.7.4 on Windows 10 operating system*

In order to find the shortest path between all cities, an algorithm was developed to generate all the possible permutations of the list of cities. The recursive function was wrapped by a stateful container that contained the shortest path currently found. The recursive function itself accepted a list of remaining cities and the currently built route. Through each recursion, the function iterated through the list of remaining cities and appended each city to the current route and removed the city from the list of remaining cities, calling the recursive function each time. This continued until the function reached a base case of having no more cities left in the list of remaining cities. At this point, the total distance of the current route would be calculated and compared with the container function’s minimum route, replacing the minimum route if a shorter path was found. In order to better optimize this solution, the starting city for every route was always set to the first city by default. Since the Hamiltonian cycle is a loop, the starting city does not matter – so long as the order of the cities is preserved, the total distance is the same. With this knowledge, the program only needed to calculate (n-1)! possible permutations with n being the number of total cities.

1. **Results** (How well did the algorithm perform?)
   1. **Data** (Describe the data you used.)

The data came in the format of uniform .tsp files with the list of city coordinates beginning on the 8th line with the x-coordinate and y-coordinates being the 2nd and 3rd strings on this line separated by white space.

* 1. **Results** (Numerical results and any figures or tables.)

|  |  |  |  |
| --- | --- | --- | --- |
| **File** | **Min Path** | **Min Distance** | **Compute Time** |
| Random4.tsp | [1-4-2-3] | 215.08553303209044 | 00:00:00.011 |
| Random5.tsp | [1-2-5-3-4] | 139.1335417499496 | 00:00:00.012 |
| Random6.tsp | [1-2-3-4-5-6] | 118.96891407553862 | 00:00:00.066 |
| Random7.tsp | [1-2-7-3-6-5-4] | 63.863031874767636 | 00:00:00.259 |
| Random8.tsp | [1-6-8-4-5-2-3-7] | 310.98207974423167 | 00:00:02.084 |
| Random9.tsp | [1-7-6-3-5-2-9-4-8] | 131.02836613987677 | 00:00:11.439 |
| Random10.tsp | [1-2-7-6-8-5-9-10-4-3] | 106.78582021866472 | 00:01:48.710 |
| Random11.tsp | [1-6-10-11-8-9-7-5-3-4-2] | 252.6844344550543 | 00:07:54.184 |
| Random12.tsp | [1-8-2-3-12-4-9-5-10-6-7-11] | 66.08484401133855 | 01:57:47.489 |

1. **Discussion** (Talk about the results you got and answer any specific questions mentioned in the assignment.)

The most notable aspect of the results is the massive increase in computational time between increasing numbers of cities, especially for 11 and 12 cities. In earlier iterations of solving this problem, the program attempted to print the results of each route onto a file, but the number of possible routes was so large that the file could not be opened due to system memory constraints. Had the realization not been reached that the starting point of the cities could be any city, the results for 12 cities could likely not have been generated in a feasible amount of time using the implemented algorithm. Using that technique to reduce the number of computations to (n-1)! greatly reduced the total execution time. This project solution clearly demonstrates the limitations of computers to analyze increasingly large datasets.

1. **References** (If you used any sources in addition to lectures please include them here.)