Project 3: Closest Edge Insetion Heuristic

Drew Stroshine
Computer Science
Speed School of Engineering
University of Louisville, USA
dmstro04@louisville.edu

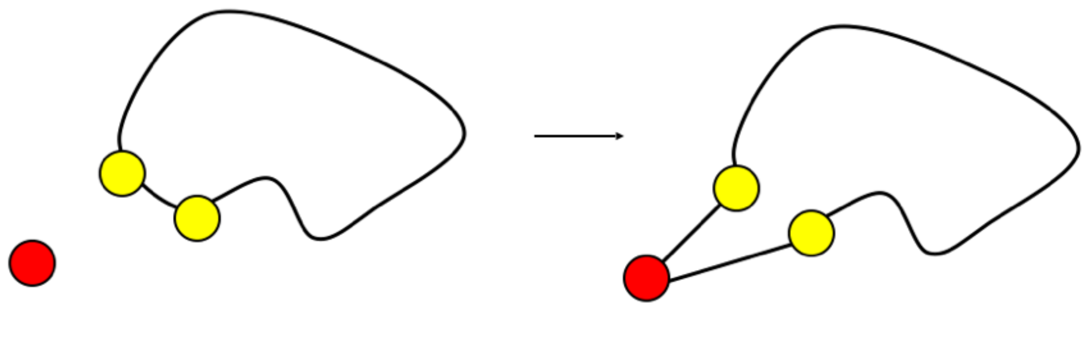
1. Introduction

In this project, I solved the Travelling Salesperson Problem by using a variant of the greedy heurisitc. This algorithm is called the closest edge insertion heuristic. The purpose for using this algorithm was to compare it's performance to previously used algorithms.

2. Approach

The algorithm I am using a variant of the greed heuristic call closest edge insertion. It does not produce the best solution, but a good one. According to the source provided in the homework instructions, it produces on average a solution 20% away from the optimal (source 1). The algorithm works by inserting nodes into the path that are closest to one of the edges already in the path. The Figure 1 below illustrates this process. In my code, I initialize the linked list ***path*** with the starting node, the nearest node in the linked list ***locs*** to the starting node, and end the path with the starting node. The starting node and the nearest node are removed from ***locs***. The I start the proces of calculating the distance between every edge in ***path*** with every node in ***locs***. The node with the shortest distance to an edge is inserted in between the two node that make up the edge and edges are created between the

Figure 1. Visual representation of closest edge insertion



new node and each edge node. The new node is removed from ***locs***, and this process continues until ***locs*** is empty. After the ***path*** is fully completed, my algorithm calculates the total distance of the ***path***. The formula used to calculate the distance between edges and nodes is the one given in the homework instructions illustrated in Figure 2 below (source 2).

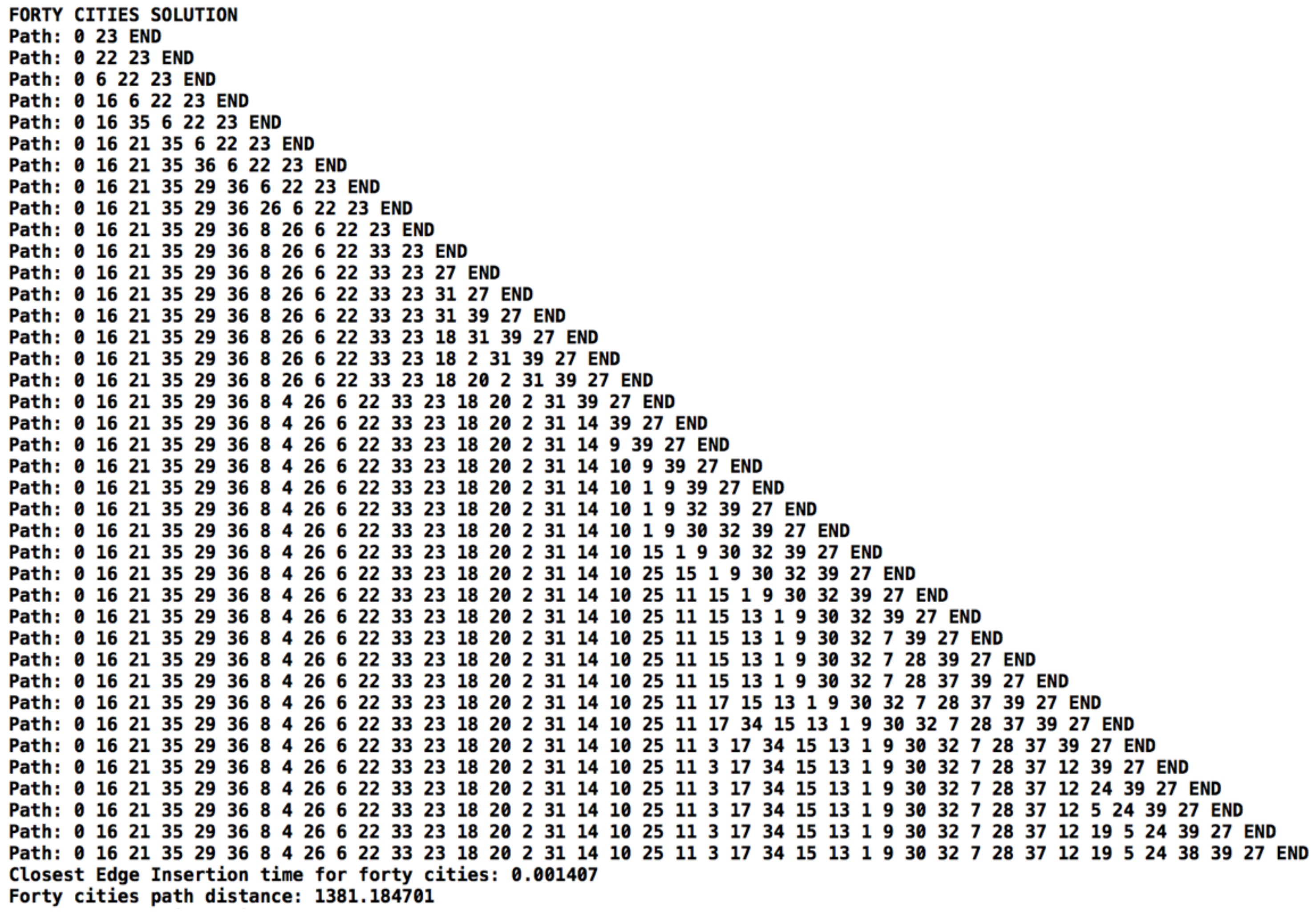
Figure 2. Formula for distance from point to line

$$d = \left|\hat{\mathbf{v}} \cdot \mathbf{r}\right| = \frac{\left|(x_2 - x_1)(y_1 - y_0) - (x_1 - x_0)(y_2 - y_1)\right|}{\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}}$$

3. Results

3.1. Data

The input data used for this project were two .tsp files called Random30.tsp and Random40.tsp. These files contained 30 and 40 cities respectively and each city was defined with an ordered pair referring to its location on a 2D cartesian graph.



```
FORTY CITIES SOLUTION
Path: 0 23 END
Path: 0 22 23 END
Path: 0 6 22 23 END
Path: 0 16 6 22 23 END
Path: 0 16 35 6 22 23 END
Path: 0 16 21 35 6 22 23 END
Path: 0 16 21 35 36 6 22 23 END
Path: 0 16 21 35 29 36 6 22 23 END
Path: 0 16 21 35 29 36 26 6 22 23 END
Path: 0 16 21 35 29 36 8 26 6 22 23 END
Path: 0 16 21 35 29 36 8 26 6 22 33 23 END
Path: 0 16 21 35 29 36 8 26 6 22 33 23 27 END
Path: 0 16 21 35 29 36 8 26 6 22 33 23 31 27 END
Path: 0 16 21 35 29 36 8 26 6 22 33 23 31 39 27 END
Path: 0 16 21 35 29 36 8 26 6 22 33 23 18 31 39 27 END
Path: 0 16 21 35 29 36 8 26 6 22 33 23 18 2 31 39 27 END
Path: 0 16 21 35 29 36 8 26 6 22 33 23 18 20 2 31 39 27 END
Path: 0 16 21 35 29 36 8 4 26 6 22 33 23 18 20 2 31 39 27 END
Path: 0 16 21 35 29 36 8 4 26 6 22 33 23 18 20 2 31 14 39 27 END
Path: 0 16 21 35 29 36 8 4 26 6 22 33 23 18 20 2 31 14 9 39 27 END
Path: 0 16 21 35 29 36 8 4 26 6 22 33 23 18 20 2 31 14 10 9 39 27 END
Path: 0 16 21 35 29 36 8 4 26 6 22 33 23 18 20 2 31 14 10 1 9 39 27 END
Path: 0 16 21 35 29 36 8 4 26 6 22 33 23 18 20 2 31 14 10 1 9 32 39 27 END
Path: 0 16 21 35 29 36 8 4 26 6 22 33 23 18 20 2 31 14 10 1 9 30 32 39 27 END
Path: 0 16 21 35 29 36 8 4 26 6 22 33 23 18 20 2 31 14 10 15 1 9 30 32 39 27 END
Path: 0 16 21 35 29 36 8 4 26 6 22 33 23 18 20 2 31 14 10 25 15 1 9 30 32 39 27 END
Path: 0 16 21 35 29 36 8 4 26 6 22 33 23 18 20 2 31 14 10 25 11 15 1 9 30 32 39 27 END
Path: 0 16 21 35 29 36 8 4 26 6 22 33 23 18 20 2 31 14 10 25 11 15 13 1 9 30 32 39 27 END
Path: 0 16 21 35 29 36 8 4 26 6 22 33 23 18 20 2 31 14 10 25 11 15 13 1 9 30 32 7 39 27 END
Path: 0 16 21 35 29 36 8 4 26 6 22 33 23 18 20 2 31 14 10 25 11 15 13 1 9 30 32 7 28 39 27 END
Path: 0 16 21 35 29 36 8 4 26 6 22 33 23 18 20 2 31 14 10 25 11 15 13 1 9 30 32 7 28 37 39 27 END
Path: 0 16 21 35 29 36 8 4 26 6 22 33 23 18 20 2 31 14 10 25 11 17 15 13 1 9 30 32 7 28 37 39 27 END
Path: 0 16 21 35 29 36 8 4 26 6 22 33 23 18 20 2 31 14 10 25 11 17 34 15 13 1 9 30 32 7 28 37 39 27 END
Path: 0 16 21 35 29 36 8 4 26 6 22 33 23 18 20 2 31 14 10 25 11 3 17 34 15 13 1 9 30 32 7 28 37 39 27 END
Path: 0 16 21 35 29 36 8 4 26 6 22 33 23 18 20 2 31 14 10 25 11 3 17 34 15 13 1 9 30 32 7 28 37 12 39 27 END
Path: 0 16 21 35 29 36 8 4 26 6 22 33 23 18 20 2 31 14 10 25 11 3 17 34 15 13 1 9 30 32 7 28 37 12 24 39 27 END
Path: 0 16 21 35 29 36 8 4 26 6 22 33 23 18 20 2 31 14 10 25 11 3 17 34 15 13 1 9 30 32 7 28 37 12 5 24 39 27 END
Path: 0 16 21 35 29 36 8 4 26 6 22 33 23 18 20 2 31 14 10 25 11 3 17 34 15 13 1 9 30 32 7 28 37 12 19 5 24 39 27 END
Path: 0 16 21 35 29 36 8 4 26 6 22 33 23 18 20 2 31 14 10 25 11 3 17 34 15 13 1 9 30 32 7 28 37 12 19 5 24 38 39 27 END
Closest Edge Insertion time for forty cities: 0.001407
Forty cities path distance: 1381.184701
```

The figures above illustrate the final route for the thirty and forty cities files. It also shows the order in which the nodes are added. The value at the end of each path titled "END" has the same coordinates as the start city, but I titled it with the word "END" so that it was easier to keep track of how my path was changing. At the bottom of the figures the time and distance values are given. For Random30.tsp, the algorithm took 0.000581 and had a total cost of 1022.581176. For the Random40.tsp, the algorithm took 0.001407 and had a total cost of 1381.184701. It took longer for the algorithm to complete the file with forty cities because there were more cities in the calculation. From my results, I can say that the algorithm produces a good solution but possibly not the best as compared to the Brute Force Algorithm, but it is quicker and actually able to hand files containing 30, 40, and easily more in a reasonable timeframe as compared to the Brute Force Algorithm which exponentially increases with each added city to the TSP problem.

5. References

Source 1: www.cs.uu.nl/docs/vakken/an/an-tsp.ppt

Source 2: http://mathworld.wolfram.com/Point-LineDistance2-Dimensional.html