**Traveling Salesperson Problem – Using A Greedy Search Heuristic**

Will B. Crawford

CECS

Speed School of Engineering

University of Louisville, USA

[wbcraw02@louisville.edu](mailto:wbcraw02@louisville.edu)

1. **Introduction**

In order to study the Traveling Salesperson Problem, or T.S.P., a greedy search heuristic was used in order to create a Hamiltonian path to find the shortest path. A greedy algorithm will bisect any given path to a city that is closest to that path. This will always ensure that with each step through the graph, any given city is always the closest. However, because this algorithm always takes the locally optimal solution, a globally optimal solution is never assured with an algorithm like this. Which means that using a greedy solution such as this, will produce an optimal path only with regards to the distance between each city.

1. **Approach**

The algorithm used is a modified version of A\* with a greedy heuristic. Instead of analyzing what node to visit with a heuristic that underestimates the distance, the algorithm simply iterates over the entire search space and finds the absolute closest city in the already generated path. In order to calculate this distance, first a perpendicular line is generated that goes through the given city to the given edge. Then, the intersection point is calculated between these two lines and then tested whether or not it falls between the two cities. If this is true, then the distance from the given city to the intersection point is calculated. Otherwise, the distance is calculated between the two cities and the algorithm will choose whichever is smallest. The algorithm will perform this distance calculation between any given city and any given edge to ensure that the truly closest city will be chosen. A previous calculation was used involving taking the projection of the city onto the line and then taking the magnitude, this however will result in a distance that is not truly closest to the given edge. A graphical user interface was developed to show the Hamiltonian path in Unity. C# was chosen as the programming language because of the ease of use and the Unity engine is easy to code in and works best with C#.

1. **Results**

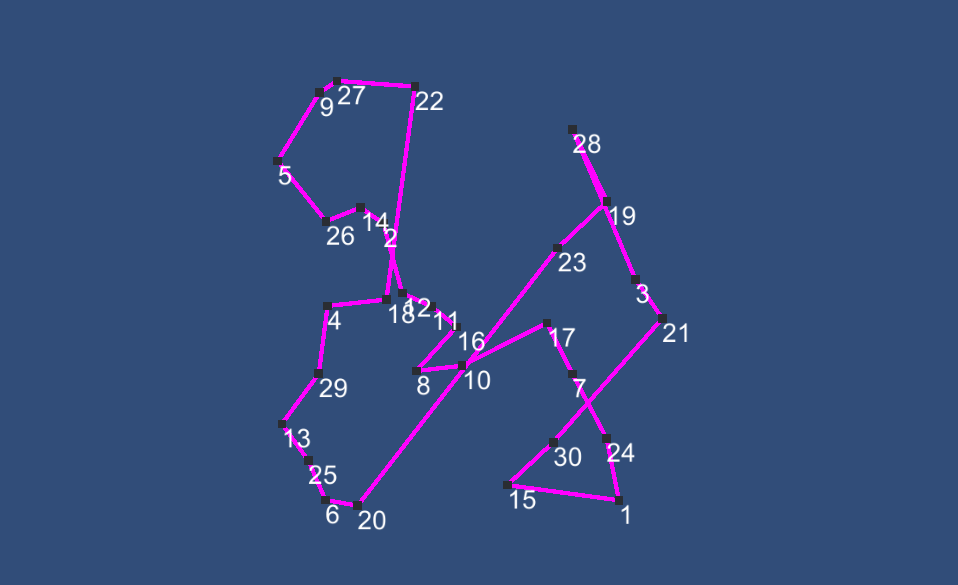
The algorithm using a greedy heuristic did not perform well when supplying it with the provided files. In *figure 1* you can see that while the path is a true Hamiltonian Path, but it has cross over edges due to implementation of the algorithm with a best path of 555.827. In *figure 4* the same problem shows itself again with a best path of 657.50. The issues from these experiments arise when adding a given vertex into the path under a certain case. This case is when the city is closer to one edge and is inserted into that edge but should be inserted into the next edge. This is due to the algorithm’s looping nature which analyzes what city is closest to each edge sequentially. So, when two edges are identical in distance, the city will be inserted into the edge that it comes to first. If you find that multiple edges tie, and you implement a tie breaker that chooses the last one it comes to, then a similar issue will arise where crossing happens. When using an external tool called Concorde, the best path generated in *figure 3* has a total weight of 517. This tool does in fact generate crossover edges. In *figure 4* the same tool generates the best path with a weight of 650. The path again, has crossover edges which again proves that greedy solutions will never contain the globally optimal solution.

* 1. **Data** (Describe the data you used.)

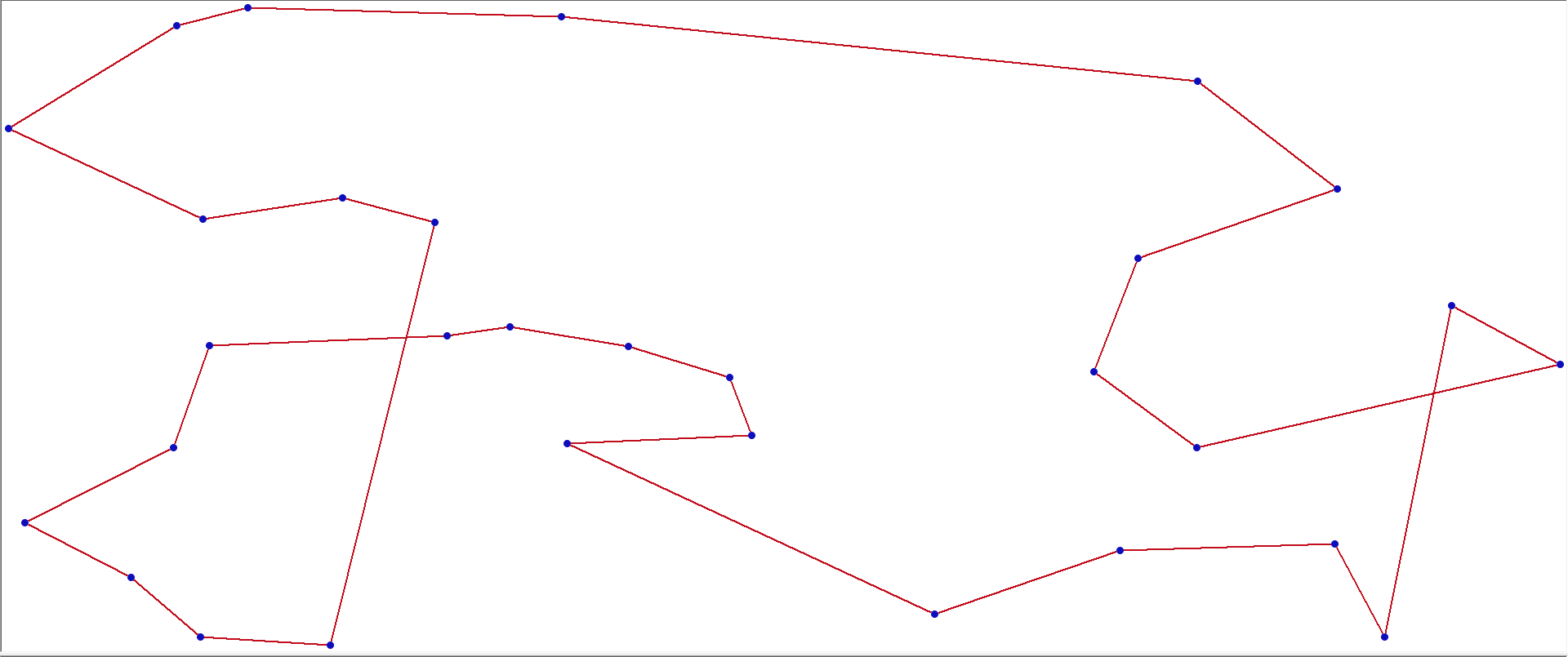
The data used was generated by a tool that randomizes Travelling Salesperson Problem nodes, the tool can be found in the references. In the following section, there are the outputs to all the given TSP files.

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

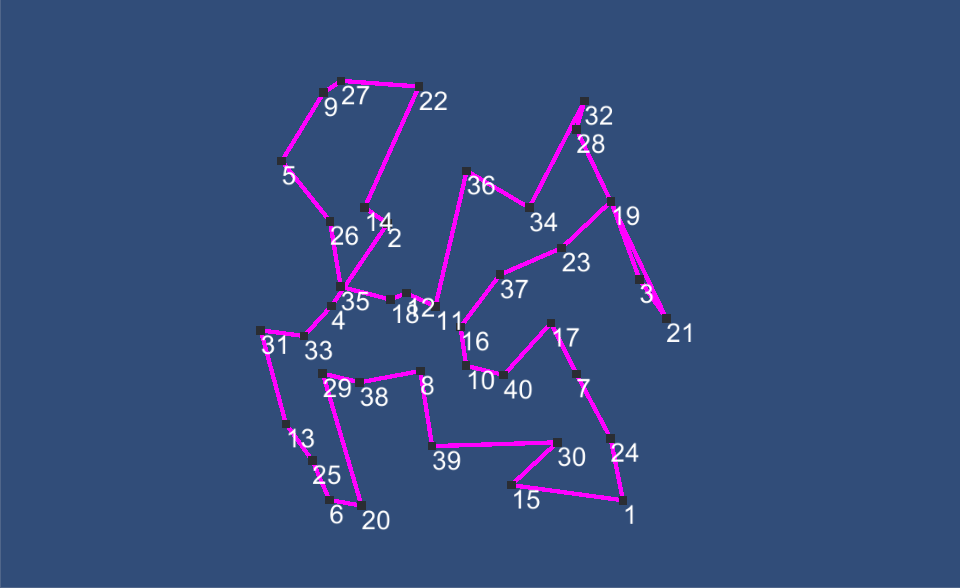
*Figure 1:* Random30.tsp Best Path



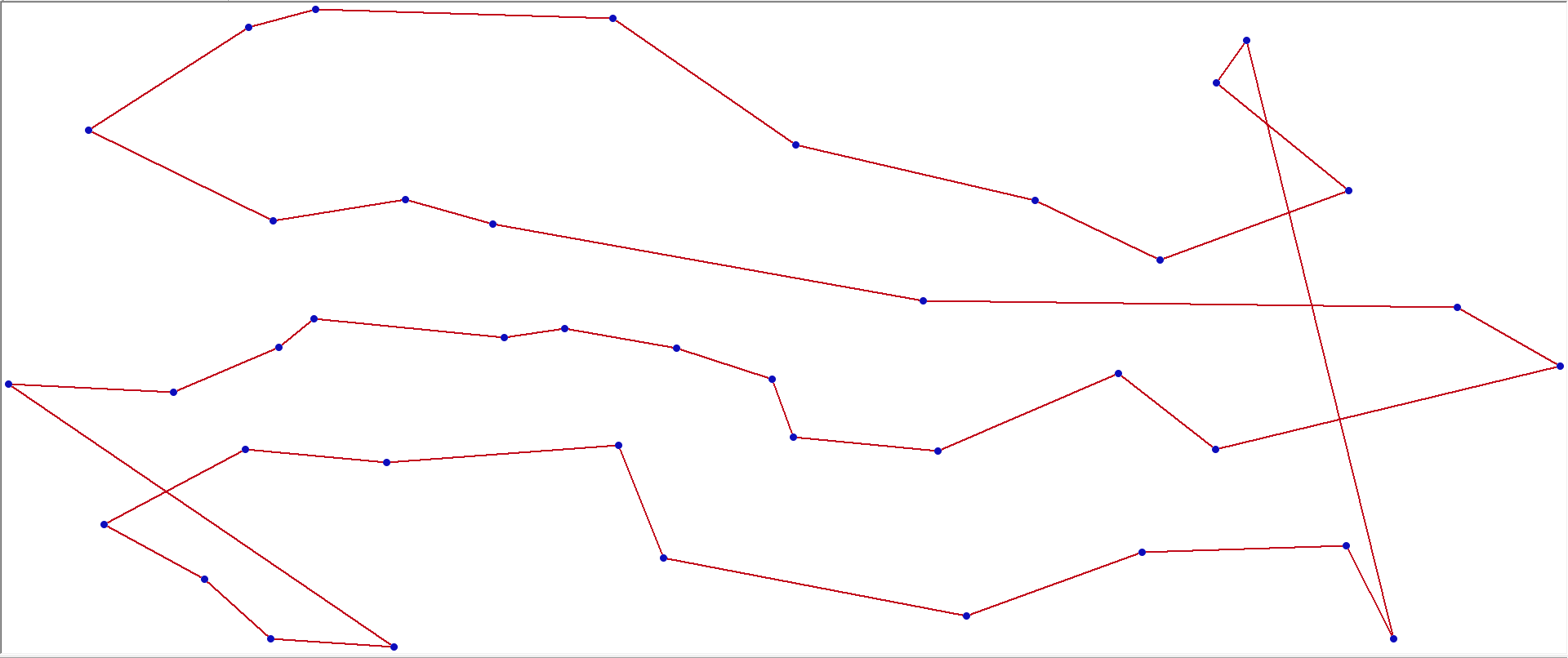
*Figure 2:* Random30.tsp Best Path using Concorde



*Figure 3:* Random40.tsp Best Path



*Figre 4:* Random40.tsp Best Path using Concorde



1. **Discussion**

The greedy-heuristic algorithm did not output the most optimal solution given the two files. This is because of cross-over and a special case when inserting a city that ties with another edge. An approach that segments the graph into multiple trivial solutions, which is a solution that contains 4 nodes, can be made and then later combined into the optimal path can be done which will generate the optimal solution using a greedy a heuristic. The algorithm used is significantly faster than brute force method because brute force would never be able to compute the total number of permutations of the Hamiltonian Path. Breadth and Depth first search cannot be compared because these algorithms can never generate a Hamiltonian Path.

1. **References**

**-** https://www.tsp.gatech.edu/concorde/index.html