EEC 289Q

March 10, 2024

Conor King

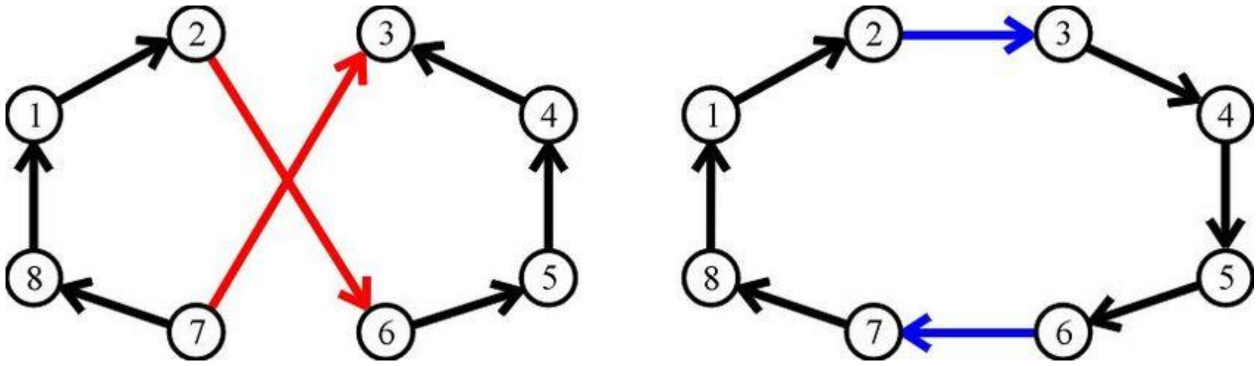
HW 3: Traveling Salesman Problem

1.

I used the same algorithm for graphs A and B.

First, I used a simple greedy algorithm. Beginning at a starting node, selecting the closest node as the next in the cycle, then repeating until all nodes have been added. I iterated over all the nodes as the start node in order to find the shortest cycle the greedy algorithm could produce.

Then I applied the 2-Opt Move method to the cycle. The 2-Opt Move iterates over every pair of edges which do not share a vertex, swaps the end points, and chooses the new arrangement if it results in a shorter cycle. In the case of edge AB and CD, it would examine a cycle with AD and CB instead. (Edge naming convention must be consistently clockwise or counterclockwise).



This image would be improved if it were undirected, but it is still the clearest illustration of how the 2-Opt Move works and improves the cycle that I could find.

Note: I really tried to find a more sophisticated algorithm which would make fuller use of the allotted 15 minutes. I started with Simulated Annealing (SA) and spent considerable time trying to tune its cooling rate. I tried to improve its performance on the Euclidean graph by starting with a greedy solution instead of an optimal one. I then found that the greedy algorithm immediately found a far better solution than the SA algorithm, and that applying SA afterword only worsened the solution! Then I tried to cluster the Euclidean graph and solve each cluster before applying the greedy algorithm, but it also performed worse. The 2-Opt Move was the only method I could find which offered a small but significant improvement over the most basic greedy algorithm.

2.

Euclidean Graph:

No. of Cycles visited: 3e6

Cost: 2543.11

Random Graph:

No. of Cycles visited: 3e6

Cost: 387.50

3.

cfking24/TSP: Travelling Salesman Problem (github.com)