Traveling Salesman Report
Kyler Volakos, ID 1210434

The Traveling Salesman Problem is an NP-Hard problem that involves finding a minimum path on a graph G, visiting all vertices exactly once before returning to the origin: it is a more general form of the Hamiltonian Cycle problem. Given the NP-Hard nature of the problem, it is not feasible typically to find the optimal answer for the problem given a large data set, so approximation algorithms are needed, using heuristics. There are a variety[1] of heuristics[2] available for approximating the TS problem, but there are trade offs involving all of them between efficiency (i.e., the ability for the heuristic to handle increasingly large datasets) and optimality/quality (i.e., the ability for the heuristic to approximate the true answer correctly).

Affecting the selection of a heuristic is the definition of the TSP itself. There are two main dividing lines affecting TSPs, whether they are metric and whether they are symmetric. In this case, we were asked to check both non-metric and metric TSPs - that is, TSPs that both obey the triangular inequality and those that do not. Metric TSPs (where the direct distance between two vertices is always the same or less than that of the distance between each of those vertices and a third vertex) are much simpler to approximate given that, as a quirk of the inequality, you are able to visit vertices multiple times, meanwhile in non-metric TSPs you cannot. Symmetric TSPs are, to my understanding, TSPs on an undirected graph, while asymmetric TSPs imply a directed graph (and thus different costs for direction of travel between two vertices). In this case, our TSPs are assumed to be symmetric (i.e., undirected).

Finally, when selecting a heuristic for each flavor of the problem, I originally had chosen to try and implement the Lin-Kernighan algorithm[3], which is a generalization of the 2-opt and 3-opt algorithms, all flavors of k-optimization which attempts to remove and replace edges in the graph to find the shortest path. While both sufficiently efficient and optimal, I found them two difficult to implement given my limited expertise, thus I instead opted to implement a minimum spanning tree for the metric TSP case, and nearest neighbor search for non-metric TSP.

In terms of optimality, minimum spanning tree and nearest-neighbor ended up more accurate for smaller graphs than the sample, and less accurate for larger graphs. Minimum spanning tree on the metric TSP was found to be less accurate than nearest-neighbor, which was surprising to me given the relative ease comparatively in solving the metric TSP, although this may be down to implementation error. Generally, however, these heuristics are likely not very accurate overall, especially with large problem sets; but their ease of implementation and, moreover, running time (nearest neighbor running in O(n^2), and minimum spanning tree running similarly in O(n^2), as both simply iterate over the graph set and sort them

---

[1] https://faculty.washington.edu/jtenenbg/courses/342/f08/sessions/tsp.html

[2] https://www.researchgate.net/publication/282245639_Heuristic_Approaches_to_Solve_Traveling_Salesman_Problem

[3] https://www.sciencedirect.com/science/article/abs/pii/S0377221710005485

appropriately). Fundamentally, the two heuristics are quite similar, save that minimum-spanning does not account for the requirement that the path not reiterate over the same node.