Kat, Quin, Nicole
December 2, 2018

Pizza Pilgrimage

The first algorithm we researched and considered was the Ant Colony algorithm. This algorithm finds the optimal path based on the idea of ants searching for food. [1] The basic concept is that when an ant finds food, it walks back to the "source" (colony home) leaving pheromones on the taken path. Based on probability, when other ants find the pheromones they are likely to follow the path. As more ants take the given path the more pheromone is laid down, and the path gets stronger. Each time food is brought back to the colony and pheromones are laid down, short paths are likely to be stronger which in turn optimizes the "solution."[1] If the path is taken less and pheromones are no longer added, the path slowly fades away. This algorithm ended up being a very sophisticated algorithm that seemed to be out of our scope for this project.

For our PP.java algorithm we used a combination of nearest neighbor and 2-opt algorithms. We chose to use this combination of algorithms because it is a simple and effective way find an optimal solution to this problem. Nearest neighbor is simply starting at a given node of the graph, and finding the node that is the shortest distance away. This is repeated for all unvisited nodes. Once all the nodes have been visited, return to the starting node. Nearest neighbor runs in $O(v^2)$ time, where v is the number of vertices in the graph. The 2-opt algorithm essentially removes 2 edges in the path and reconnects them in a different way. If after the change in edges has occurred and the new path is shorter than the previous path, then it becomes the "optimal" solution. Otherwise, it is discarded.[2] So in this section of the program, we first found the best solution when using the nearest neighbor algorithm. Once we had found that path,

---

[1] Macura, Wiktor K. "Ant Colony Algorithm." From *MathWorld*--A Wolfram Web Resource, created by Eric W. Weisstein. http://mathworld.wolfram.com/AntColonyAlgorithm.html

[2] Andy. "Technical-Recipes.com." *Technicalrecipescom*, 15 June 2017, www.technical-recipes.com/2017/applying-the-2-opt-algorithm-to-traveling-salesman-problems-in-java/.

we then ran it through the 2-opt algorithm to determine the overall best path. The running time of the 2-opt algorithm is also $O(v^2)$, where v is the number of vertices in the graph. The overall running time for this algorithm is $O(2v^2)$, which equals $O(v^2)$, where v represents the vertices in the graph.

The triangle inequality states that the distance from A to B to C is never shorter than going from A to C.[3] Thus, the direct path from one node to the next is always shorter than visiting another node before the final node. To solve this problem, we ran a depth first search traversal over a minimum spanning tree. We chose to use this algorithm because it can be implied that the given graph can be created in Euclidean space and distance. Because of this, based on the Special Cases: TSP with Triangle Inequality section of Karkory and Abudalmola's research paper, one of the best ways to find an optimal solution is to use the idea of a minimum spanning tree.[3] To do this we started with a random point in the graph. We then found the smallest edge that connects the starting vertex to the next vertex. This is the start of our minimum spanning tree (MST). Next, select the smallest edge from all edges connecting to the MST. Continuously repeat that process, making sure there are no loops, until there are no more nodes to add to the MST. Finally, run a depth first traversal of the created tree to construct the shortest path. The running time to create the MST is O(elogv), were e represents the edges, and v represents the vertices. The running time of a depth first search traversal over a minimum spanning tree is O(v), where v represents the vertices. So, the overall running time of this

[3] Karkory, Fatma A, and Ali A Abudalmola. "World Academy of Science, Engineering and Technology International Journal of Computer and Information Engineering." *Implementation of Heuristics for Solving Travelling Salesman Problem Using Nearest Neighbour and Minimum Spanning Tree Algorithms*, 10 Nov. 2013

algorithm is O(v + elogv), which simplifies to O(elogv), where e represents the edges, and v represents the vertices.