**Description of Algorithm**

For solving the given “travelling salesman” problem, finding the shortest path between fully interconnected nodes, we used an ant colony optimization algorithm with a randomized, greedy heuristic within local space.

The ant colony optimization essentially runs on two different parameters. We send out 5 ants from 5 random nodes that move to new nodes to create a path based on a combination of the inverse probability heuristic and a broad scale heuristic that is dependent on previous calculations of the shortest path, called a pheromone value. The greedy heuristic probabilistically chooses shorter nodes in order to create a path. The ant colony optimization technique records the best paths chosen over time and adds a pheromone value to each edge to route ants in the direction of the pheromone (shorter paths).

The inverse probability solely depends on the shortest remaining paths an ant can take. The calculation is (1) a sum over the distances between the current node and the current available nodes, followed by (2) a division of each distance value over the sum, then taking (3) the inverse of the previously calculated percentages, summing these values, and again (4) dividing each value by the sum. The shortest remaining path will have the highest probability to be chosen (proportionally) while the longest remaining path will have the lowest.

The pheromone is calculated over time and only after each full path calculation is completed. Each of the ants will have completed a path through all the nodes. We then give each of the taken paths pheromone values depending on how successful they were, where the shortest path gets the highest pheromone value. For the first iteration the pheromone value is the same for all nodes and the path only depends on the probability heuristic. For successive iterations, all edges in the pheromone table decay slightly over time (mentioned in greater detail later).

To select the next node for an ant, we multiply the greedy values for all the nodes by the pheromone value for all the nodes and put them into a list that is normalized into a probability mass function. We then choose our next node by using a sampling function (sample\_from\_pmf) over the pmf, which returns an index into our list of current nodes to visit. After each step, we update the path, distance travelled, and the remaining valid nodes for each ant.

During the run of the algorithm, after each iteration, the pheromone value of the nodes has to gradually be reduced, as it would only stack up over time otherwise and have an impact on the result even though it is no longer part of the optimal path. Therefore, we multiply the pheromone value of each node every iteration of the algorithm with a set scalar to make sure the pheromone “evaporates” over time.

The decay value is controlled by a scale and is very small (~0.995), but, over 10000 iterations, this plays a large role in the growth of many explored paths in the initial iterations and the exploration around a single, strong path in later iterations.

Lastly, we only record the calculated minimum distance and path if they are smaller then the current minimum values. In addition, since the ant path starts from a random node (to help with overarching convergence of a path given the same starting position), we recover the path put into the output by starting it from node 1. Therefore, all paths will appear as if they started from node 1.

Our recorded average (and values) over 10 iterations (also in readme):

Average: 27635.9

Values:

27625

27625

27779

27625

27616

27625

27616

27616

27616

27616