Metaheuristics TP4 – Ant Search

Ning, Tien-Tso

Introduction

The goal is to implement the Ant Search algorithm (AS), and compare the algorithm to the previous implemented Simulated Annealing (SA), as well as the baseline Nearest-Neighbor (Greedy) algorithm. The problem that the algorithms attempt to solve is the "Traveling Salesman Problem" (TSP). The problem is generalized as a list of cities, and the goal is to find the best tour (shortest path length) of the all the cities, starting and ending at the same city (completing the tour). The Ant Search algorithm takes its inspiration from how ants search and optimize their movements when searching for food by using pheromones and leveraging the idea of collective work to produce a solution.

Methodology

There are a couple of parameters that, defined beforehand, will make the explanation of the algorithm easier to follow. The parameters are as follows: $\alpha, \beta, \rho, Q, \pi_0$. Corresponds to constants α, β , the evaporation factor, and Q, π_0 obtained from the NN-Algorithm in order to get a baseline for the quantity of each ant's pheromone laid on their tour, as well as the initial pheromone starting on each edge of the graph, respectively. The constants α, β are chosen empirically, and determine the influence of the pheromone laid by ants, as well as the influence of visibility (the heuristic information ants have on how far a city is). These constants are empirically found, but intuitively help us understand that pheromone influence is exponential rather than linear. The evaporation factor is important since it is one of the factors that allows the poorer-solutions to fade away over time, allowing for exploitation. In our experiment, the evaporation factor was one-tenth, meaning that after each time-step, only nine-tenths of the pheromone originally laid by ant k remains on the paths. Altering this evaporation factor by decreasing or increasing the value could aid in exploration and exploitation, respectively. However, one must be careful not to alter the value too much as it can negatively impact the quality of the algorithm's solution or the run-time to obtain an acceptable solution. For instance, having no evaporation factor would lead to absolutely no exploration, and the search would quickly become a decision of which path was first taken by the first ant. But having an almost-instantaneous evaporation would cause the search to become stochastic as each ant takes a random path. We can liken this evaporation factor to the same feeling of the temperature change in SA. We want to gradually reduce the temperature, just as we want to gradually let the pheromone fade over time in order to balance exploration/exploitation of our search space. The quantity of pheromone and the initial pheromones for the edges of the graph are determined by running the greedy algorithm (NN-Algorithm) ten times and getting the average path length. This is done to get a good baseline for how long a baseline solution (path) should be, allowing us to get constants that fit the problem-space better. We can liken this as setting the initial temperature in SA, to get a feel for what the problem search-space looks like in order to fine-tune parameters that are expected to give our algorithm the best chance at success.

The Ant Search algorithm runs as follows: At each time step, a set of m ants runs the graph, and generates their individual solution. The individual ants k generate their solution by starting at a random city i, and moves to the next city j such that j has not been visited before. City j is accepted as a movement depending on the equation:

$$p_{ij}^{k}(t) = \begin{cases} \frac{(\tau_{ij}(t))^{\alpha}(\eta_{ij})^{\beta}}{\sum_{l \in J}(\tau_{il}(t))^{\alpha}(\eta_{il})^{\beta}} & \text{if } j \in J \\ 0 & \text{otherwise} \end{cases}$$

This equation takes into account the amount of pheromone on the paths, as well as the inverse distances of the path (which is considered heuristic information of visibility). Once all m ants have created their solution, the graph is updated with their pheromone trails, and the next timestep begins. After the max timesteps have been reached, the algorithm returns the best solution found during the algorithm's lifetime. This is expected to be the global optimal solution (or close enough to the global optimal solution).

Results

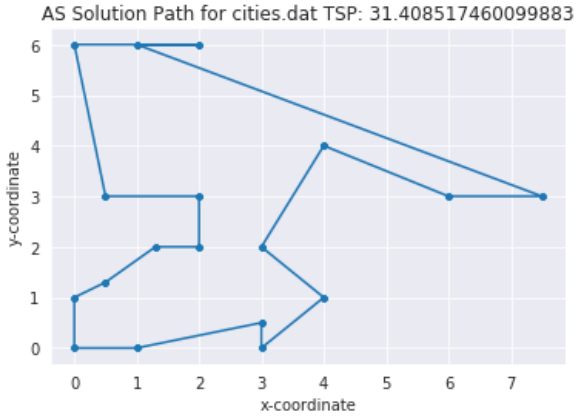
We chose the best results from ten runs of the AS against the NN algorithm (averaged over ten runs) and provide the visuals.

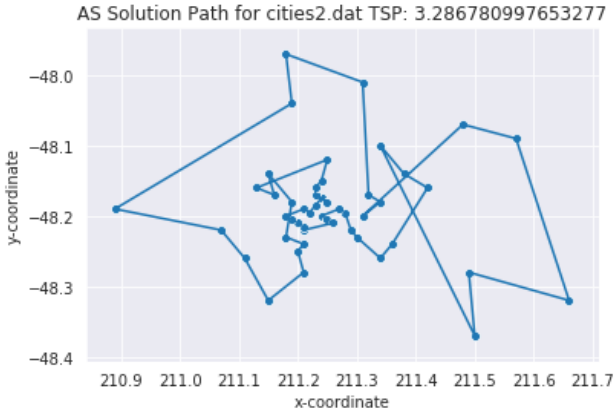
cities.dat results: (['k', 'o', 'r', 'e', 'l', 'd', 'j', 'n', 'b', 'a', 'c', 'f', 'h', 'i', 'q', 'g', 'p', 'm', 'k'], 31.408517460099883)

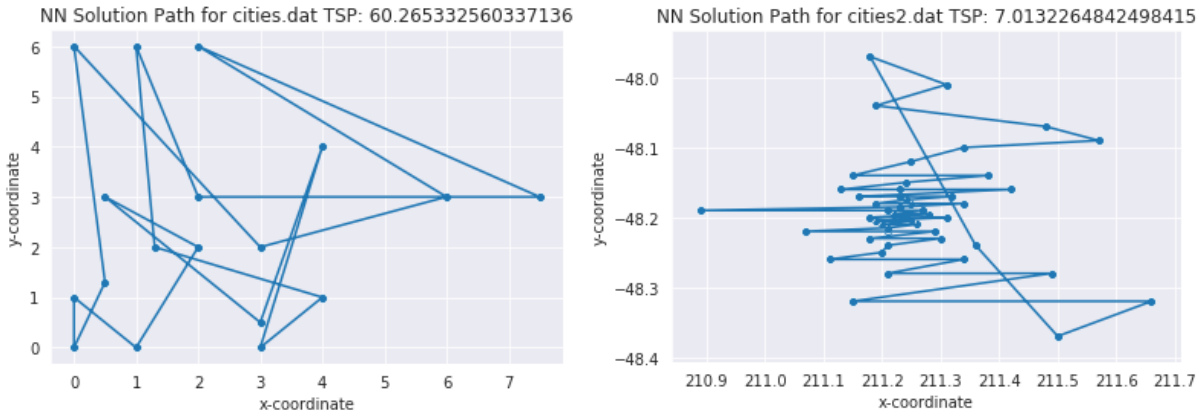
cities.dat NN results: 65.50684116422799

cities2.dat results: (['c15', 'c12', 'c10', 'c21', 'c17', 'c19', 'c28', 'c31', 'c33', 'c24', 'c26', 'c29', 'c37', 'c39', 'c41', 'c44', 'c16', 'c20', 'c09', 'c13', 'c04', 'c05', 'c48', 'c49', 'c46', 'c06', 'c02', 'c01', 'c03', 'c08', 'c11', 'c14', 'c18', 'c23', 'c25', 'c34', 'c36', 'c40', 'c42', 'c45', 'c47', 'c43', 'c35', 'c38', 'c30', 'c27', 'c32', 'c07', 'c22', 'c15'], 3.29125227139956)

cities2.dat NN results: 7.24813502988272







50-cities

cities_50_1.dat results: (['c18', 'c35', 'c24', 'c29', 'c19', 'c7', 'c14', 'c23', 'c36', 'c6', 'c43', 'c46', 'c38', 'c45', 'c2', 'c40', 'c31', 'c20', 'c26', 'c49', 'c8', 'c37', 'c32', 'c33', 'c34', 'c22', 'c10', 'c0', 'c17', 'c12', 'c21', 'c28', 'c5', 'c1', 'c15', 'c39', 'c25', 'c13', 'c16', 'c4', 'c41', 'c27', 'c47', 'c44', 'c42', 'c30', 'c48', 'c3', 'c11', 'c9', 'c18'], 1678.2953765600348)

cities_50_2.dat results: (['c37', 'c1', 'c42', 'c24', 'c27', 'c22', 'c38', 'c25', 'c11', 'c47', 'c40', 'c43', 'c10', 'c17', 'c39', 'c49', 'c3', 'c12', 'c13', 'c36', 'c46', 'c41', 'c8', 'c4', 'c35', 'c18', 'c0', 'c2', 'c23', 'c21', 'c30', 'c14', 'c6', 'c48', 'c33', 'c28', 'c44', 'c7', 'c16', 'c15', 'c9', 'c26', 'c19', 'c32', 'c5', 'c31', 'c20', 'c34', 'c29', 'c45', 'c37'], 2014.988617205455)

cities_50_3.dat results: (['c0', 'c23', 'c38', 'c2', 'c26', 'c7', 'c4', 'c32', 'c18', 'c36', 'c37', 'c28', 'c43', 'c42', 'c45', 'c25', 'c14', 'c40', 'c21', 'c33', 'c3', 'c10', 'c49', 'c20', 'c27', 'c11', 'c13', 'c12', 'c15', 'c35', 'c46', 'c22', 'c1', 'c44', 'c41', 'c39', 'c16', 'c34', 'c5', 'c29', 'c17', 'c31', 'c47', 'c9', 'c8', 'c30', 'c19', 'c48', 'c24', 'c6', 'c0'], 1672.1816488953477)

cities_50_4.dat results: (['c27', 'c9', 'c7', 'c30', 'c11', 'c21', 'c46', 'c4', 'c39', 'c3', 'c47', 'c38', 'c42', 'c22', 'c25', 'c31', 'c13', 'c6', 'c33', 'c15', 'c40', 'c10', 'c48', 'c2', 'c34', 'c45', 'c35', 'c20', 'c32', 'c12', 'c29', 'c26', 'c44', 'c16', 'c41', 'c18', 'c8', 'c23', 'c24', 'c19', 'c14', 'c49', 'c28', 'c5', 'c1', 'c0', 'c36', 'c37', 'c17', 'c43', 'c27'], 1492.056993144261)

cities_50_5.dat results: (['c28', 'c3', 'c29', 'c26', 'c22', 'c42', 'c31', 'c14', 'c23', 'c49', 'c47', 'c2', 'c20', 'c19', 'c5', 'c30', 'c0', 'c45', 'c36', 'c4', 'c24', 'c21', 'c40', 'c44', 'c18', 'c32', 'c12', 'c43', 'c35', 'c38', 'c37', 'c46', 'c10', 'c16', 'c41', 'c15', 'c25', 'c8', 'c13', 'c27', 'c33', 'c7', 'c48', 'c6', 'c1', 'c34', 'c39', 'c11', 'c17', 'c9', 'c28'], 1606.4938210556943)

cities_50.dat NN results: 5359.401511639641

Mean Execution Time: 40.374

Standard Deviation: 13.787

60-cities

cities_60_1.dat results: (['c52', 'c50', 'c49', 'c12', 'c43', 'c8', 'c36', 'c55', 'c32', 'c17', 'c26', 'c0', 'c27', 'c56', 'c19', 'c47', 'c34', 'c4', 'c7', 'c14', 'c45', 'c16', 'c22', 'c10', 'c53', 'c51', 'c21', 'c37', 'c15', 'c18', 'c57', 'c5',

'c40', 'c24', 'c30', 'c29', 'c39', 'c13', 'c44', 'c9', 'c2', 'c48', 'c58', 'c31', 'c1', 'c11', 'c42', 'c38', 'c46', 'c20', 'c35', 'c33', 'c6', 'c3', 'c54', 'c23', 'c41', 'c28', 'c25', 'c59', ‘c52’], 1925.470527758225)

cities_60_2.dat results: (['c11', 'c52', 'c39', 'c59', 'c38', 'c36', 'c42', 'c28', 'c54', 'c44', 'c16', 'c31', 'c34', 'c50', 'c41', 'c43', 'c14', 'c58', 'c48', 'c12', 'c4', 'c47', 'c22', 'c35', 'c56', 'c32', 'c7', 'c30', 'c25', 'c21', 'c40', 'c29', 'c3', 'c51', 'c19', 'c2', 'c18', 'c33', 'c5', 'c55', 'c9', 'c20', 'c15', 'c49', 'c53', 'c17', 'c6', 'c8', 'c57', 'c13', 'c0', 'c24', 'c10', 'c46', 'c37', 'c45', 'c23', 'c26', 'c1', 'c27', ‘c11’], 1782.1354274906887)

cities_60_3.dat results: (['c0', 'c56', 'c53', 'c32', 'c13', 'c3', 'c10', 'c26', 'c5', 'c15', 'c18', 'c7', 'c6', 'c30', 'c36', 'c40', 'c55', 'c27', 'c12', 'c54', 'c21', 'c14', 'c28', 'c39', 'c29', 'c34', 'c41', 'c38', 'c25', 'c42', 'c51', 'c47', 'c59', 'c31', 'c45', 'c37', 'c57', 'c58', 'c49', 'c2', 'c24', 'c46', 'c52', 'c50', 'c22', 'c19', 'c17', 'c33', 'c1', 'c11', 'c4', 'c48', 'c9', 'c20', 'c23', 'c8', 'c16', 'c35', 'c44', 'c43', ‘c0’], 1557.400445105938)

cities_60_4.dat results: (['c14', 'c18', 'c6', 'c54', 'c12', 'c27', 'c9', 'c24', 'c43', 'c4', 'c25', 'c33', 'c40', 'c5', 'c50', 'c49', 'c10', 'c30', 'c39', 'c45', 'c41', 'c29', 'c3', 'c46', 'c59', 'c52', 'c37', 'c55', 'c26', 'c8', 'c48', 'c53', 'c21', 'c19', 'c0', 'c57', 'c15', 'c31', 'c23', 'c58', 'c7', 'c22', 'c16', 'c11', 'c20', 'c42', 'c17', 'c35', 'c56', 'c1', 'c47', 'c34', 'c32', 'c2', 'c13', 'c38', 'c36', 'c51', 'c44', 'c28', ‘c14’], 1550.506506145416)

cities_60_5.dat results: (['c2', 'c15', 'c43', 'c52', 'c26', 'c10', 'c4', 'c13', 'c6', 'c9', 'c49', 'c47', 'c48', 'c59', 'c53', 'c33', 'c58', 'c42', 'c18', 'c14', 'c41', 'c7', 'c44', 'c50', 'c21', 'c55', 'c54', 'c35', 'c23', 'c37', 'c12', 'c5', 'c38', 'c51', 'c24', 'c11', 'c16', 'c32', 'c30', 'c1', 'c0', 'c40', 'c31', 'c3', 'c19', 'c20', 'c34', 'c25', 'c45', 'c27', 'c39', 'c56', 'c28', 'c57', 'c46', 'c17', 'c36', 'c29', 'c8', 'c22', ‘c2’], 2172.1163450374233)

cities_60.dat NN results: 6761.3058919495

Mean Execution Time: 697.144

Standard Deviation: 61.112

80-cities

cities_80_1.dat results: (['c11', 'c46', 'c69', 'c18', 'c41', 'c70', 'c9', 'c15', 'c37', 'c48', 'c14', 'c63', 'c60', 'c16', 'c52', 'c1', 'c36', 'c53', 'c75', 'c28', 'c72', 'c34', 'c29', 'c5', 'c54', 'c77', 'c71', 'c35', 'c58', 'c33', 'c51', 'c65', 'c59', 'c30', 'c74', 'c57', 'c21', 'c47', 'c3', 'c12', 'c78', 'c62', 'c32', 'c39', 'c67', 'c23', 'c76', 'c8', 'c38', 'c17', 'c26', 'c49', 'c22', 'c43', 'c4', 'c64', 'c6', 'c56', 'c27', 'c20', 'c73', 'c13', 'c40', 'c68', 'c61', 'c31', 'c42', 'c2', 'c7', 'c24', 'c19', 'c25', 'c10', 'c0', 'c45', 'c79', 'c44', 'c50', 'c66', 'c55', ‘c11’,], 1972.065825180104)

cities_80_2.dat results: (['c40', 'c50', 'c33', 'c20', 'c39', 'c68', 'c69', 'c41', 'c67', 'c6', 'c43', 'c45', 'c4', 'c78', 'c72', 'c24', 'c26', 'c74', 'c1', 'c77', 'c42', 'c13', 'c16', 'c64', 'c14', 'c60', 'c51', 'c62', 'c59', 'c46', 'c76', 'c54', 'c15', 'c19', 'c12', 'c9', 'c10', 'c21', 'c32', 'c34', 'c18', 'c29', 'c66', 'c8', 'c73', 'c65', 'c35', 'c52', 'c0', 'c22', 'c25', 'c61', 'c5', 'c79', 'c36', 'c3', 'c57', 'c2', 'c49', 'c44', 'c30', 'c31', 'c56', 'c75', 'c63', 'c70', 'c53', 'c23', 'c48', 'c71', 'c58', 'c47', 'c17', 'c11', 'c37', 'c27', 'c55', 'c28', 'c7', 'c38', ‘c40’], 1866.3836921147774)

cities_80_3.dat results: (['c15', 'c32', 'c33', 'c19', 'c7', 'c46', 'c37', 'c27', 'c54', 'c18', 'c45', 'c70', 'c4', 'c38', 'c75', 'c55', 'c6', 'c34', 'c9', 'c69', 'c72', 'c59', 'c44', 'c30', 'c68', 'c26', 'c39', 'c17', 'c23', 'c1', 'c71', 'c63', 'c42', 'c47', 'c2', 'c57', 'c60', 'c31', 'c16', 'c40', 'c5', 'c11', 'c61', 'c58', 'c52', 'c21', 'c24', 'c29', 'c50', 'c65', 'c43', 'c76', 'c49', 'c22', 'c20', 'c62', 'c53', 'c10', 'c12', 'c28', 'c8', 'c3', 'c67', 'c36', 'c14', 'c64', 'c79', 'c73', 'c25', 'c78', 'c51', 'c77', 'c13', 'c0', 'c56', 'c74', 'c35', 'c41', 'c48', 'c66', ‘c15’], 2001.6047504632481)

cities_80_4.dat results: (['c6', 'c48', 'c78', 'c10', 'c1', 'c75', 'c24', 'c76', 'c49', 'c12', 'c14', 'c26', 'c62', 'c30', 'c79', 'c53', 'c4', 'c23', 'c77', 'c13', 'c17', 'c73', 'c41', 'c65', 'c16', 'c63', 'c46', 'c11', 'c20', 'c37', 'c18', 'c59', 'c22', 'c50', 'c71', 'c45', 'c31', 'c35', 'c34', 'c27', 'c52', 'c36', 'c33', 'c15', 'c74', 'c7', 'c56', 'c28', 'c2', 'c72', 'c47', 'c51', 'c54', 'c66', 'c32', 'c38', 'c21', 'c25', 'c61', 'c69', 'c64', 'c29', 'c9', 'c60', 'c3', 'c68', 'c5', 'c57', 'c39', 'c0', 'c43', 'c40', 'c70', 'c42', 'c67', 'c8', 'c44', 'c55', 'c19', 'c58', 'c6'], 2218.342916862223)

cities_80_5.dat results: (['c3', 'c52', 'c41', 'c66', 'c4', 'c13', 'c47', 'c18', 'c64', 'c32', 'c57', 'c5', 'c74', 'c73', 'c70', 'c22', 'c7', 'c61', 'c63', 'c0', 'c6', 'c69', 'c30', 'c27', 'c49', 'c77', 'c65', 'c23', 'c53', 'c44', 'c20', 'c45', 'c75', 'c55', 'c33', 'c58', 'c50', 'c26', 'c12', 'c56', 'c48', 'c39', 'c59', 'c79', 'c17', 'c2', 'c68', 'c1', 'c14', 'c11', 'c9', 'c34', 'c15', 'c10', 'c16', 'c24', 'c28', 'c54', 'c78', 'c60', 'c42', 'c46', 'c71', 'c51', 'c72', 'c40', 'c25', 'c35', 'c38', 'c62', 'c21', 'c8', 'c37', 'c36', 'c43', 'c76', 'c19', 'c67', 'c29', 'c31', 'c3'], 1954.1911489182598)

cities_80.dat NN results: 8992.68267140849

Mean Execution Time: 1642.345

Standard Deviation: 133.944

100-cities

cities_100_1.dat results: (['c20', 'c2', 'c77', 'c0', 'c18', 'c6', 'c56', 'c73', 'c3', 'c9', 'c74', 'c61', 'c28', 'c98', 'c15', 'c8', 'c71', 'c25', 'c50', 'c69', 'c21', 'c14', 'c94', 'c34', 'c81', 'c42', 'c62', 'c84', 'c68', 'c55', 'c63', 'c33', 'c49', 'c89', 'c26', 'c13', 'c70', 'c96', 'c40', 'c92', 'c27', 'c95', 'c97', 'c36', 'c75', 'c66', 'c64', 'c88', 'c99', 'c37', 'c59', 'c83', 'c22', 'c17', 'c51', 'c80', 'c52', 'c46', 'c7', 'c58', 'c87', 'c48', 'c12', 'c90', 'c30', 'c93', 'c24', 'c41', 'c23', 'c10', 'c67', 'c31', 'c44', 'c60', 'c86', 'c19', 'c45', 'c72', 'c54', 'c4', 'c11', 'c47', 'c38', 'c29', 'c85', 'c82', 'c53', 'c39', 'c32', 'c91', 'c57', 'c1', 'c35', 'c43', 'c16', 'c78', 'c5', 'c79', 'c76', 'c65', 'c20'], 2401.386033041816)

cities_100_2.dat results: (['c52', 'c92', 'c91', 'c3', 'c43', 'c49', 'c98', 'c64', 'c46', 'c31', 'c48', 'c11', 'c84', 'c26', 'c55', 'c0', 'c17', 'c65', 'c29', 'c16', 'c13', 'c89', 'c94', 'c69', 'c20', 'c59', 'c5', 'c8', 'c44', 'c7', 'c23', 'c19', 'c78', 'c85', 'c34', 'c81', 'c72', 'c83', 'c82', 'c53', 'c30', 'c21', 'c93', 'c36', 'c76', 'c96', 'c33', 'c57', 'c80', 'c1', 'c58', 'c22', 'c75', 'c2', 'c87', 'c39', 'c27', 'c42', 'c63', 'c10', 'c14', 'c61', 'c6', 'c50', 'c56', 'c73', 'c38', 'c62', 'c97', 'c68', 'c45', 'c18', 'c77', 'c74', 'c12', 'c54', 'c9', 'c60', 'c67', 'c28', 'c79', 'c4', 'c15', 'c41', 'c90', 'c66', 'c70', 'c32', 'c35', 'c37', 'c71', 'c95', 'c99', 'c86', 'c47', 'c88', 'c40', 'c24', 'c25', 'c51', 'c52'], 2389.410898256839)

cities_100_3.dat results: (['c53', 'c7', 'c40', 'c35', 'c17', 'c31', 'c25', 'c74', 'c50', 'c16', 'c58', 'c51', 'c59', 'c47', 'c62', 'c95', 'c85', 'c46', 'c72', 'c44', 'c78', 'c81', 'c24', 'c4', 'c98', 'c14', 'c8', 'c37', 'c67', 'c86', 'c23', 'c55', 'c75', 'c1', 'c82', 'c63', 'c93', 'c88', 'c18', 'c5', 'c71', 'c39', 'c90', 'c61', 'c38', 'c83', 'c49', 'c27', 'c42', 'c64', 'c70', 'c45', 'c69', 'c0', 'c10', 'c36', 'c33', 'c99', 'c15', 'c87', 'c6', 'c68', 'c73', 'c9', 'c32', 'c96', 'c91', 'c84', 'c19', 'c60', 'c3', 'c54', 'c12', 'c41', 'c34', 'c77', 'c92', 'c22', 'c57', 'c52', 'c30', 'c48', 'c66', 'c26', 'c76', 'c11', 'c28', 'c56', 'c29', 'c21', 'c43', 'c89', 'c97', 'c65', 'c2', 'c79', 'c94', 'c20', 'c80', 'c13', 'c53'], 2356.4175037894356)

cities_100_4.dat results: (['c6', 'c96', 'c73', 'c71', 'c16', 'c28', 'c33', 'c58', 'c1', 'c9', 'c92', 'c70', 'c95', 'c3', 'c0', 'c80', 'c89', 'c38', 'c98', 'c39', 'c56', 'c35', 'c52', 'c40', 'c55', 'c54', 'c60', 'c63', 'c75', 'c57', 'c66', 'c17', 'c44', 'c65', 'c90', 'c81', 'c4', 'c84', 'c45', 'c20', 'c49', 'c25', 'c61', 'c72', 'c5', 'c97', 'c64', 'c77', 'c91', 'c47', 'c46', 'c32', 'c79', 'c15', 'c19', 'c51', 'c14', 'c67', 'c12', 'c8', 'c21', 'c87', 'c78', 'c18', 'c30', 'c88', 'c85', 'c22', 'c7', 'c34', 'c68', 'c82', 'c23', 'c93', 'c31', 'c59', 'c10', 'c24', 'c29', 'c26', 'c41', 'c50', 'c86', 'c99', 'c37', 'c48',

'c94', 'c43', 'c69', 'c83', 'c13', 'c11', 'c74', 'c36', 'c53', 'c2', 'c62', 'c42', 'c76', 'c27', 'c6'], 2407.119074993301)

cities_100_5.dat results: (['c67', 'c82', 'c89', 'c59', 'c30', 'c29', 'c17', 'c38', 'c84', 'c73', 'c18', 'c26', 'c58', 'c62', 'c54', 'c3', 'c50', 'c66', 'c72', 'c32', 'c12', 'c36', 'c74', 'c39', 'c7', 'c16', 'c0', 'c90', 'c91', 'c37', 'c63', 'c55', 'c78', 'c46', 'c14', 'c27', 'c24', 'c68', 'c80', 'c76', 'c8', 'c21', 'c86', 'c42', 'c9', 'c52', 'c11', 'c57', 'c47', 'c45', 'c64', 'c79', 'c15', 'c51', 'c1', 'c56', 'c5', 'c25', 'c85', 'c49', 'c43', 'c23', 'c88', 'c99', 'c75', 'c13', 'c97', 'c20', 'c10', 'c35', 'c4', 'c92', 'c31', 'c98', 'c65', 'c6', 'c83', 'c61', 'c2', 'c77', 'c28', 'c22', 'c87', 'c81', 'c33', 'c70', 'c96', 'c53', 'c93', 'c69', 'c95', 'c19', 'c41', 'c40', 'c48', 'c34', 'c60', 'c94', 'c44', 'c71', 'c67'], 2127.1308295918607)

cities_100.dat NN results: 9573.587177661402

Mean Execution Time: 3906.416

Standard Deviation: 242.501

Discussion

Averaged over a couple of runs, we chose m to match the number of cities in the problem-space (one city for each ant), and the t_{max} value to be small enough to have a decent run-time. We found that this compromise of parameters provided good solutions in a reasonable amount of time, since we wanted enough ants to explore the problem-space, but over a reasonable amount of iterations such that the run-time of the algorithm did not take too long as problem-space sizes increased. We noticed that as we increased the number of ants m to match the number of cities, that the quality of our solutions improved and we noticed that when we increased the number of iterations, the more stable our solutions were (more likely to produce good results). This is supported by the empirical results from the original AS algorithm, as they noticed that it took more than twenty ants to bias a path, and that pheromone influence is exponential rather than linear.

On a ten-run averaged Nearest-Neighbor solution, the AS algorithm provides an improvement of solution path-length by a factor of two (a solution that is twice as good). Showing that on average, our algorithm finds an optimal-solution that is twice as good, indicating that our algorithm does a decent job at exploring the search space, and seeking out a solution close to if not the global optima.

On our generated TSP problems, the AS algorithm outperforms the NN algorithm by over a factor of two in most cases, with the best runs beating the NN algorithm by providing a solution three-times as good. This is expected since the NN algorithm greedily chooses the closest path, which as the problem search-space increases, causes poorer and poorer solutions.

In regards to the run-time on our generated TSP problems, as the problem-size increased, we also increased the number of ants (without increasing the number of iterations in order to preserve lower run-times, as discussed above). As the number of cities increased, the time it took to execute the algorithm also increased. This is natural since the number of ants to create solutions will increase the number of calculations. For the most part, the algorithm is relatively stable, and execution times don't vary too much (standard deviation).

Comparing AS and SA algorithms, in terms of the quality of solutions, the AS seems to find comparably better solutions more often, delivering overall better quality and stability. This may simply be due to our

implementation differences, as the SA algorithm still outperforms the NN algorithm (even though it does not outperform at the same scale as the AS algorithm does). However, the execution times of the SA is exceptionally faster than the AS algorithm. This can be mostly attributed to the fact that we increased the number of ants to match the problem search-space (number of cities) which drastically increased run-time, while in SA, the run-time is only controlled by number of iterations and number of acceptances/tries before the temperature is dropped. In AS, because of the nature of collective-work, more calculations must be made and taken into account, and thus the run-time should be expected to be higher than in SA when all other factors are comparable.

Conclusion

In conclusion, our Ant Search algorithm optimizes a solution path that performs at least twice as well as the baseline Nearest-Neighbor algorithm, and compares favorably to the Simulated Annealing algorithm presented in the previous TP.