Homework 10: Heuristic Algorithms

Data Structures and Algorithms Spring 2020

Sparsh Bansal

1 Question 1

The greedy algorithm that I have implemented starts from a given starting index (or defaults to the first city in the input instance case) and then sequentially adds the next closest city in terms of the given edge weights to the tour. Towards the end of the algorithm, I add the starting city to the tour in order for the salesman to reach the starting city after the tour. Specifically, the algorithm uses a dictionary with Boolean mappings to track whether it has added the city to the tour or not, and continually adds the closest city to the variable *path* and adds the respective edge weight (or distance) to the variable *dist*.

This is a greedy approach because it makes decisions based on the immediate cases, and does not record the long-term effects of the decisions. For example, it chooses the last city of the tour without considering its distance to the starting city, which would not be optimal in a good number of cases.

2 Question 2

The local search algorithm that I have implemented starts by procuring an output from the greedy heuristic algorithm described in question 1. It then sweeps through the path, and tries to find a more optimized path, by swapping two consecutive cities per loop. If it finds one or more relatively optimized solutions, it is stored in the variables *minpath* and *mindist*, and prints them out in the terminal for the salesman.

This is a local search heuristic algorithm because it tries to use an existing path and optimize it locally, by making a change locally and determining if the changed path is better optimized or not.

3 Question 3

In order to compare the results of the algorithms, I have recorded the runtimes (RT) and optimality gaps (O. Gap) in the table below.

Performance Table				
Dataset	GHA (RT in μS)	LSHA (RT in μS)	GHA (O. Gap)	LSHA (O. Gap)
gr17	355.874	117.923	4.4 %	3.1 %
gr21	489.817	160.148	10.9 %	10.9 %
gr24	603.958	273.943	22.1 %	20.1 %
gr24	2016.633	782.525	16.3 %	15.3 %

I observe that the runtimes for both the GHA (Greedy Heuristic Algorithm) and the LSHA (Local Search Heuristic Algorithm) are increasing proportionally with the length of the data-set (or, the number of cities). It is also visible that the LSHA takes lesser time to finish looping, than the GHA on average, which is expected, since the LSHA is dependent on making smaller, local changes to observe increase in optimality.

In terms of the optimality gap, the LSHA either finds a more optimal solution to the TSP, or returns the initial path, which is why according to the table, the optimality gap never increases for the LSHA on the same data-set. The optimality gap is generally increasing along with the length of the data-set, apart from the 48 city data-set on which the heuristic algorithms perform better than the 24 city data-set which is off-trend.