Daniel Fernández
932-063-474
ME538
17 November 2014
Homework #3

The Travelling Salesman Problem (TSP) is a common combinatorial optimization problem. The premise of the TSP is to find the optimal Hamiltonian path when given a list of n cities, making sure each city is visited at the least possible cost. Solutions are provided by a number of algorithms, with many losing capability as n increases. As will be shown, a linear increase in n will provide an exponential increase in calculation cost. From a general standpoint, the simulation will seek to minimize the following relation:

$$Travel\ Cost = min \sum_{i=1}^{n} \sum_{j=1}^{n} d_{i,j} x_{i,j}$$

Where (i, j) are 2-dimensional coordinates, d is the added distance between nodes, and x is the tour, or collection of vectors. In each simulation, two lists of cities are compared, one of 10 and one of 20 cities. The possible routes of travel is related as a factorial expression, or

$$Possible\ Routes = (n-1)!$$

So in the case of 10 cities, the possible outcomes is 9!, or 362,880 routes. In the case of 20 cities, the possible outcomes is 19!, or 1.22 x 10^{17} routes. This obviously becomes a computational strain rather quickly, and the use of heuristics, or logical estimates is recommended to reduce the cost. The first A* heuristic examined is a simple and straightforward greedy one, the Nearest Neighbor approach. The premise is to always select the shortest distance from node to node. This is a quick but not necessarily optimal algorithm as it does not factor potentially better combinations of routes. The other heuristic employed is the Kruskal Algorithm, which spans all possible routes. This algorithm takes all edges, or paths and sorts them in an increasing order. By adding edges and repetitively checking for a minimal path, the optimal path can be found. This approach is more thorough, but more computationally exhaustive, and as such is more time-consuming.

In addition, an Evolutionary Algorithm (EA) is applied to the TSP and presented as an alternative to the A* search method. In a similar manner as the biological world, an EA collects populations, mutates them, measures their values, and then weeds out any undesirables. As is shown, Evolutionary Algorithms provide tempting solutions because though not as thorough as other methods, EA's can provide near-optimal solutions at a fraction of the computational cost. The solution presented uses a random initialization scheme which selects populations or routes at random and removes those with the larger travel costs.

Simulation Results:

The results are shown below in Figures 1 – 6. Of immediate note are the mirrored results for all 3 (n = 10) city cases. This makes sense with fewer nodes and less spacing; there is a higher chance that the simple greedy distance is also the least cost distance. This concept breaks down though as more cities are introduced. In figure 2, the simple greedy path shows a higher cost as the agent tends to "corner" itself on the map. The ending position of node 20 is also not ideal, as there is a high cost to return to home city 1. Figure 4 shows the Kruskal solution and is regarded as the most complete. Here the agent corners itself only when necessary and with a lower-cost circular path. The ending node 20 is in a much more ideal location closer to city 1. Figure 6 shows the Evolutionary Algorithm and deserves appropriate attention. Though following a different path than in Figure 4, the EA still shares many similar low-cost trajectories with the Kruskal as well as a decent terminal node. As mentioned earlier, the EA proves the most effective: providing near-optimal solutions at a computational cost several orders of magnitude lower than the A* Solution.

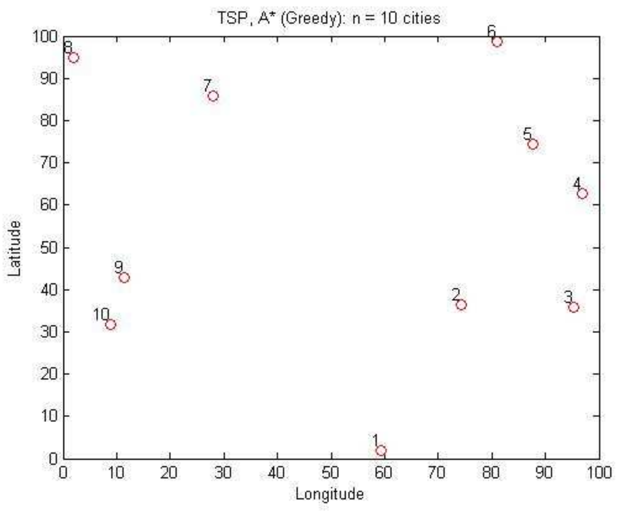

Figure 1

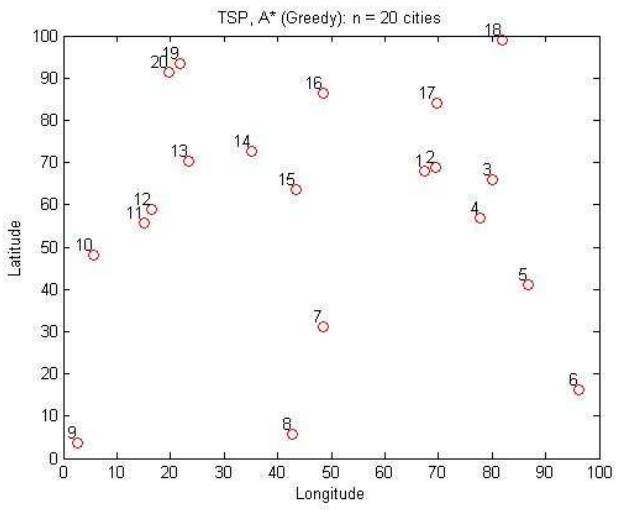

Figure 2

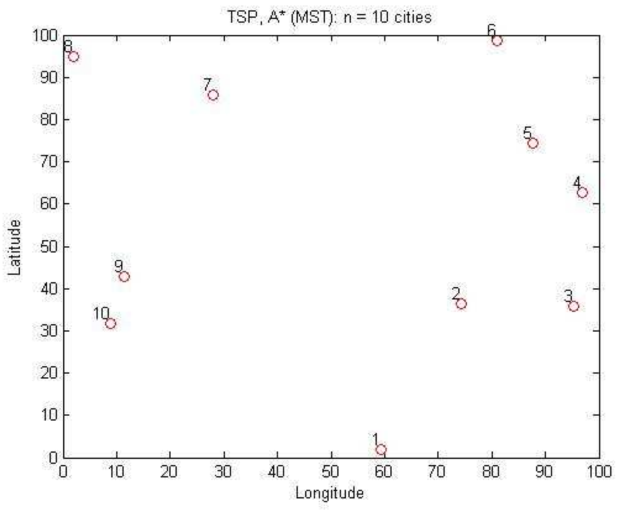

Figure 3

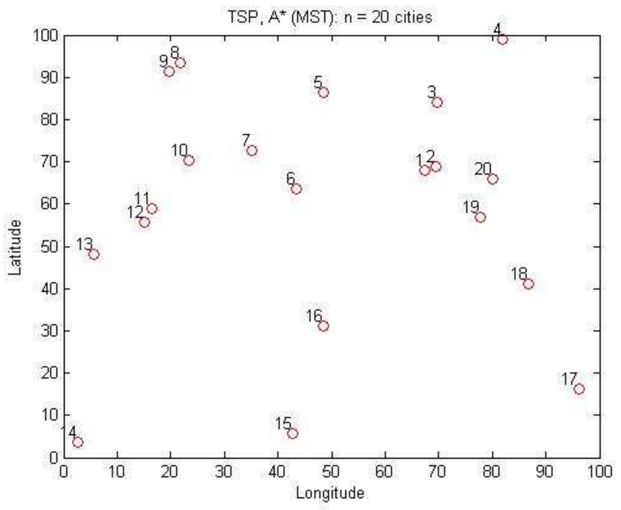

Figure 4

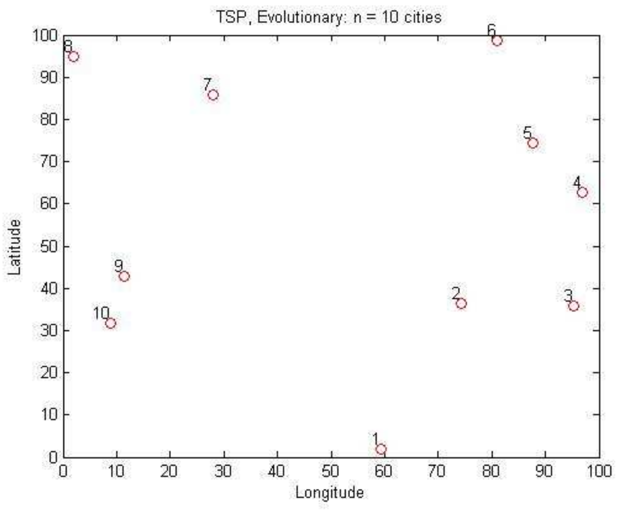

Figure 5

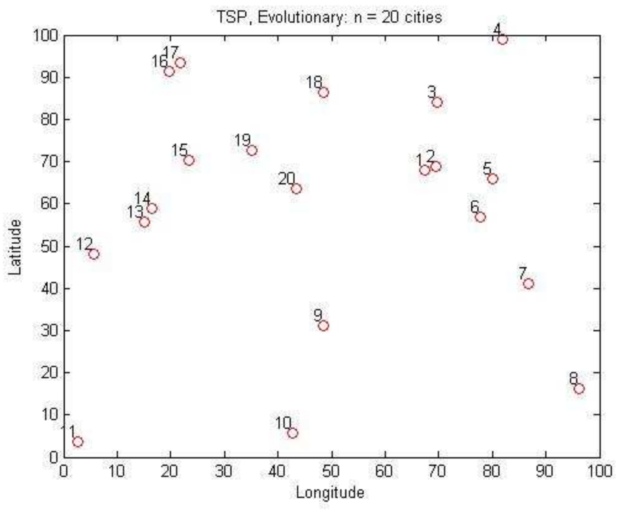

Figure 6