Oregon State University

Project Group 21

CS 325: Project 4

Charles Jenkins Albert Le Colin Bradford

June 7, 2015

1 The Ideas Behind the Algorithm

Our program makes use of two different TSP algorithms in tandem to achieve better optimization for the tests. We start with a greedy algorithm and then feed its results into a 2-opt algorithm. We then take the results from the 2-opt algorithm and feed it back into the 2-opt algorithm over and over again until we either no longer find an improved route or we reach some time limit.

The whole process works as follows:

- Read the input file and build a set of coordinates for each city.
- Calculate the distances for every possible edge in the graph.
- Build a 2D list of the edges as (distance, source, destination) triplets. Since this would be a symmetrical square nxn matrix, we only create the lower half of it to save processing time. We also do not include edges with a cost of zero (edges from a node to itself).
- Sort this list in ascending order of distances.
- Pass this list to the greedy algorithm.
- Pick the lowest cost edge.
- Add its source to our route, mark it as visited, and increase by one the degree of its source and destination vertices.

- While we haven't visited all the nodes, add to our route the next lowest cost edge whose source matches our previous edge's destination, ensuring that no vertex ever has a degree of more than two and that we form no cycles. Return the distance and route.
- Feed that resulting route into the 2-opt algorithm.
- Perform standard 2-opt swaps (reorder routes such that they do not cross over themselves), keeping track of any better solutions we find until either we find no more improvements or we reach some time limit. Return the distance and route.
- Continue feeding the resulting route of the 2-opt algorithm back into the 2-opt algorithm until either we find no more improvements or reach some time limit.
- Output to file our final distance and route solution.
- NOTE: On very large input sizes, like in `tsp_example_3.txt`, we resort to only a 2-opt solution on a random starting path to ensure reasonable termination times (execution can be stopped midway and still offer a solution, unlike the greedy algorithm which requires a fully built and sorted table up front).

Acknowledgments for the basic ideas behind each algorithm can be found in the source code.