CS325 Winter 2013: Implementation 3 Report

Daniel Reichert
Trevor Bramwell

March 17, 2013

Approach Overview

In our approach to the TSP we came up with a series of steps that allow the problem to be broken into smaller pieces, parallelized and optimized locally before being combined into a complete tour. The first step is to run k-means to produce k distinct subsets of the cities that are localized. This produces neighborhoods that effectively divides the program into small enough parts such that an optimal solution is more easily found. It is also a prerequisite to allowing a parallelized algorithm.

Next, for each neighborhood we ran the greedy nearest neighbor algorithm to produce an initial neighborhood tour which we would later optimize.

After acquiring the optimal tour that nearest neighbor can produce, we run 2-opt on the tour to exhaustion. According to Johnson and McGeoch [table 10] 2-opt can produce results that are within 5 percent of the optimal tour of the graph.

Once 2-opt is run on the tour, it is necessary to connect each of the neighborhoods together in order to create a single tour of all of the cities. One at a time the neighborhoods are merged together until a complete tour of all cities is established.

Steps in Detail

K-means++ K-means++ needs to be passed a graph and k, where k is the number of clusters that will be created. It starts by choosing a node a random from the graph. Once a node is selected it then computes the node furthest from itself, then the node furthest from the first and second. This continues till k nodes are chosen. Then each node in the graph (with the exception of the nodes chosen for k) associates itself with the nearest cluster.

Nearest neighbor Nearest neighbor starts at a given node and then finds the closest node to itself and visits it. As each node is visited is subsequently removed from the pool of nodes that are considered for the next node. Before further optimizing the tour that is found by nearest neighbor, we run the nearest neighbor process repeatedly. Because the nearest neighbor algorithm is cheap to run, we are able to find the minimal length tour that it can produce by considering every possible node as the starting point.

2-OPT 2-OPT is a local search heuristic that replaces two edges in a TSP graph with two cheaper edges. It can either be ran once, or until the cheapest local option is found. The 2-OPT we implemented was the exaustive search for the cheapest option.

Connecting Neighborhoods The process of connecting the neighborhoods together involves figuring out what order the neighborhoods should be connected in. This is accomplished by repeating the nearest neighbor and 2-opt process, but instead of running it on a subgraph of the problem, the neighborhoods are treated as a unit. In order to treat them as a single unit, we find the center of each neighborhood by finding the average x and y value and considering that for a node. With the order in which the neighborhoods should be glued together established, a method that is very similar to the swap function used in 2-opt is used to merge each neighborhood tour into a single tour.

Implementation

While our approach had great promise, its complexity proved to be too much given our allowed time frame to implement it. There were a small number of bugs and false assumptions which prevented us from using each piece in the solution that we submitted.

The most important false assumption that we made was that each point would be distinct. Because in our minds no graph that someone would want to apply a TSP algorithm to would have duplicate points, we did not consider that possibility in our algorithms. However some of the test and example input contained duplicate points, and as such broke our algorithms in unexpected ways. This took an inordinate amount of time to debug, and prevented us from using all of the functions that we had implemented all at once.

The result of running out of time meant that the best we could submit was the solution from the nearest neighbor algorithm with the input of the optimal starting node.

Resources Used

The Traveling Salesman Problem: A Case Study in Local Optimization. By David S. Johnson and Lyle A. McGeoch. November 20, 1995. `http://www2.research.att.com/~dsj/papers/TSPchapter.pdf`

The Traveling Salesman Problem (TSP). By Rahul Simha. The George Washington University `http://www.seas.gwu.edu/~simhaweb/champalg/tsp/tsp.html`

Pattern Recognition and Machine Learning. By Christopher M. Bishop. p424-450. 2006.

Algorithms. By Dasgupta, Papadimitriou, and Vazirani. p283-305. July 18, 2006.