TSP Report (CS325 Project 4)

By Eric Rouse; a Team of 1.

The Traveling Salesperson Problem

I tried a couple of approaches to this problem. I tried reducing the TSP to a linear problem set, but couldn't really figure out how to apply any bounds. Then I implemented a very simply greedy algorithm that didn't work at all. That is when I came across the Christofides Algorithm. I found some decent documentation, so I gave it a shot. My implementation is still not perfect; there is some iteration to do to make it faster and more robust, but that will have to wait. It will be my spring break project.

Christofides Algorithm

The algorithm can be stated in plain English as:

1. Create a minimum spanning tree (MST) from the list of cities.
2. Take all the vertices in the MST and make a perfect matching tree (PMT)
3. Combine the MST and the PMT.
4. Tour the combined tree (CMT) using the Eulerian tour technique (ETT) and create a circuit.
5. Make the ETT circuit visit each vertex only once by skipping visited nodes.

My Implementation

I used Python 3 to implement those steps in a function I called tsp(). The main routine also handles the command line argument of the input file, reads the input and instantiates the signal handler. After that the main routine calls tsp() on the list of nodes that was created from the input file. Nodes are a class designed to hold the name of the node "self.n", as well as the x and y position of the city "self.x" and "self.y".

The tsp() function takes the cities list and tries to apply the Christofides Algorithm:

1. First it creates a minimum spanning tree for the list, using mst(); mst() is an implementation of Prim's Algorithm to find a minimum spanning tree.
2. Then it takes every edge and creates its reverse for perfect matching. So it finds an edge that went from u to v and creates an edge from v to u. Thus every vertex has a match.
3. The lists created in step 1 and 2 are joined. Now an Eulerian tour is possible.
4. So, it calculates the tour using the ett() function; ett() returns an Eulerian circuit on a given MST.
5. The tour is then traversed and any repeat edges are removed.
6. The tour length is calculated and compared to the best found so far. If the new length is better, that length and path are stored. To satisfy the "stop on command" requirement the best path found and best path length found (so far) are stored in global variables after each iteration of tsp(). Upon receipt of the kill signal, these values are written to a file.
7. This sequence is repeated for every possible combination of MST.

Other Important Functions

mst() – Minimum Spanning Tree

The mst() function uses Prim's algorithm to generate the MST for a given list of cities. Starts out with a single vertex and adds each closest city to the MST and removes that city from the given list. When the city list is empty, the MST has been fully generated.

ett() – Eulerian tour technique

Takes in an edge list and starts on the first node (called the source). It looks for edges from that node and traverses down the first one, removing it from the edge list. It continues this until the edge list is empty. This is one place for massive optimization; finding an algorithm for picking the best path for a given MST.