Name: User-ID: mbkb74

Algorithm A: Greedy

Algorithm B: Christofides

Description of enhancement of Algorithm A:

For my enhancement of greedy I looked at the times where there were multiple minimums to choose between. My original algorithm just chose the first occurrence, however in my enhanced version I implemented additional functions to look ahead and see what impact choosing a different minimum would have.

The function test_next_step looks at what would happen if you chose the tour with a given node appended to it. It calculates the full route that would be taken. When choosing this route it has an element of recursion as when choosing the route, the program will come across more times where there are multiple minimums. In order to make the program terminate in a timely manner though, there is an additional parameter, called depth, which restricts the depth of recursion. It calls index_decision to choose which index it should choose when met with a decision.

The function index_decision takes a list of indices and a tour, and generates the tours with the indices appended and calls test_new_step with each new tour. In this it also increases the value of depth to keep track.

Description of enhancement of Algorithm B:

The final step of Christofides algorithm is to convert from an Eulerian circuit into a Hamiltonian circuit by skipping repeated vertices. In my initial implementation I did this by looping over the tour with duplicates, and if the vertex wasn't in my new list of a tour without duplicates, then add it,

However this does not give an optimal solution as the tour length changes depending on which repeated vertex is chosen to be left in. So for my improvement I found all the vertices that were repeated in the Eulerian circuit and for each one, compared the effect of keeping the vertex in each position on the length of the tour. The one that gave the shortest tour stayed and the rest were removed. I then repeated this for all the repeated vertices until the tour had no duplicates.