1.1 Introduction

Travelling salesman problem is the problem where a set of cities and distance between every pair of cities are given, the problem is to find the shortest route that visits every city exactly once and returns to the starting point. If there exist a tour that visit every city exactly once is known as Hamiltonian tour. We have to find minimum Hamiltonian cycle. The problem is a famous NP-hard problem.

Nearest neighbour algorithm:

This algorithm is greedy in nature used to find minimum route among all cities.

Step 1:An arbitrary city is chosen as initial vertex, mark as current vertex

Step 2: Find out the shortest edge connecting the current vertex and an unvisited vertex.

Step 3: Mark current vertex as visited one.

Step 4:Then next unvisited vertex marked as current vertex and repeat step 2.

Step 5: This process continues till all nodes are visited.

Time complexity for nearest neighbour algorithm is O(n^2).

1.2 Problem statement

Travelling Salesman Problem: Given a set of cities and distance between every pair of cities, and the shortest possible route that visits every city exactly once and returns to the starting point.

1.3 Methodology

The principle used behind this algorithm is nearest neighbour with 2-opt to get the shortest possible route to travel all cities.

If the coordinates given by user is Euclidean then distance is calculated between coordinates by formula $\mathbf{d = \sqrt{[(x_2 - x_1)^2 + (y_2 - y_1)^2]}}$.

If the coordinates given by user is Non-euclidean then distance is calculated between coordinates by formula $\mathbf{d = [(x_2 - x_1)^2 + (y_2 - y_1)^2]}$.

The basic nearest neighbour algorithm used as

It starts with index 0 city and add the city to solution list .Then with remaining cities it select the nearest distance between its neighbour and starting city and add it to solution .It find nearest distance between newly added city to solution list and its neighbour .This process is repeated until all cities are visited. At the end, if all cities are visited then it add the distance between last visited city and starting point to the total distance.

My original approach was to find the best starting point by repeatedly selecting different starting point and finding the best route but it didn't always result in the best improved route.

1.4 Improvement

To improve the route I used 2-opt exchange method where the endpoints of two edges of a tour are swapped and if the resulting tour has a lower distance.The algorithm iterates through each city on the

tour. And perform 2-opt exchange and if tour distance is shorter than previous one then only it will retains the change.

To further reduce the 2-opt algorithm I used triangle inequality to determine if a swap will reduce distance ,with this I reduced the number of exchanges .It stops when distance does not gets improved further.

Further improvement in time complexity of 2-opt algorithm can be made by maintaining a neighbour list of each city and sorted by increasing distance of neighbours from original city.It will help for fewer iterations.

1.5 Conclusion and Future scope

I can conclude that it finds shortest route to travel among all cities.

For future scope further improvement in time complexity of 2-opt algorithm can be made by maintaining a neighbour list of each city and sorted by increasing distance of neighbours from original city. It will help for fewer iterations.