AI PROJECT REPORT

Travelling Salesman Problem (TSP)

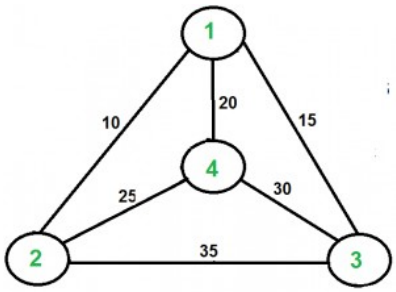
Team: -

Team Members	Roll number
Vani Agarwal	18ucs098
Hritwik Singhal	18ucs055
Abhay Singhal	18ucs011
Sameer Gupta	18ucs008

Contents

Introduction

Traveling Salesman Problem (Statement):
Given a set of cities and distance between every pair of cities, the problem is to find the shortest possible route that visits every city exactly once and returns to the starting point.



Ex, In the graph shown in figure above. a TSP tour in the graph is 1-2-4-3-1. The cost of the tour is 10+25+30+15 = 80.

This is a NP hard problem so There is no polynomial time know solution for this problem.

We can solve it in many different ways:

1. Brute force method: we have to generate all permutations taking every city one by one as the starting point and flag the minimum of them and that will be our solution.
2. 2-opt method: a simplified form of the Lin-Kernighan algorithm. It is basically a tour improvement method which takes a given tour and attempts to modify it in order to obtain an alternative tour of lower cost
3. Genetic Algorithm (GA): GA is based on the mechanics of biological evolution and is one of the algorithms that have found extensive use in solving optimization problems
4. Particle Swarm Optimization: The PSO is originally attributed toJames Kennedy and RussellEberhart [23]and was intended tosimulatesocial behavior involved in the movement of flock of birds.
5. K-means clustering: This is the process of partitioning n points into k sets on the basis of their spatial distribution.The main idea is to define kcentroids, one for each cluster, in such a way that they are placed as far away from each other as possible [25]. The next step is to take each point in the data set and associate it to the nearest centroid. Then, re-calculate knew centroids for each of the clusters resulting from the previous step. This is repeated until no more changes are made.

Algorithm applied

We have solved the TSP using **Minimum Spanning tree Heuristic**.

The MST has been implemented using **Prim's Algorithm**.

The input is in the form of **Adjacency List of edges**.

Algorithm:
1) We will take node 0 as starting and ending point for our problem.
2) Then we will use Prim's Algorithm to Construct MST (taking node 0 as root) of the graph.
3) Finally we will traverse the visited vertices in pre-order method of the constructed MST to get one of optimal solution and add node 0 at end to complete the walk.

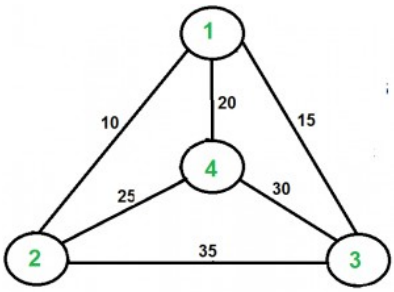
Experiments

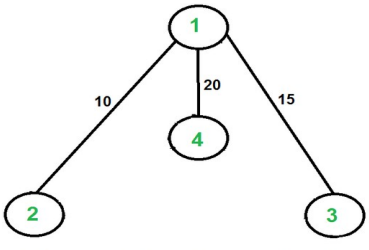
a) Study and solve it using minimum-spanning-tree (MST) heuristic

Minimum Spanning Tree

A spanning tree of a graph is a sub-graph that contains all the vertices of the graph and is a tree. a graph may have many spanning trees; and the minimum spanning tree is the one with the minimum sum of the edge costs of the tree.

Triangle inequality is a theory according to which the shortest path between two vertices is the their direct connection rather than going through some other node so using this concept we can solve the above problem better than the brute force approach of dynamic programming approach.





Minimum Spanning Tree of the graph on left side with 1 as root. The Preorder Traversal of MST is 1-2-4-3. So the output is 1-2-4-3-1

Considering the diagram where first diagram is our map and the second diagram is its implementation with MST.

Here we can see it that implementation with MST will give us a path as of a bit optimal solution then brute force or dynamic bu its pre-order traversal here for this example this is the best solution that is 1-2-4-3-1 with a cost of 80 which in this case is optimal but not necessary in every case.

To confirm that this is better solution than brute force or dynamic we define a term ***full walk.*** a full walk is lists all vertices when they are first visited in pre-order, it also list vertices when they are returned after a subtree is visited in pre-order. The full walk of above tree would be 1-2-1-4-1-3-1.

Proofs:-

- The total cost of full walk is at most twice the cost of MST as we traverse every edge twice at most.

- Because we are printing just the pre-order of the above tree we can say that the output will obviously less than Full path as well as in pre-order, two or more edges of full walk are replaced with a single edge.
- Best cost of possible TSP can never be less than the cost of MST as the MST of a graph is the minimum cost tree that connects all vertices.

HEURISTICS

- Every edge in the graph is placed in a priority queue, giving priority to the lowest-costing
- edge.
- We use Prim's algorithm to create a minimum spanning tree. This is done by arbitrarily picking a start node and using the priority queue to repeatedly add the current lowest cost edge connected to the tree, one edge at a time, until all vertices are included.
- This minimum spanning tree is transformed into an eulerian tour by doubling each of the edges.
- The eulerian tour is transformed into an aTSP solution by traversing its edges, replacing each edge that would visit an already visited node with an edge to the next node along the eulerian tour that hasn't yet been visited. This results in an aTSP solution because each node will be visited only once, and it will end at the start node because that is what the eulerian tour already does. To ensure a solution is found in incomplete graphs, our algorithm repeats steps 2 through 4 with each city as a starting node for the MST (up to 50 cities) and returns the best solution found.

b) Find and implement an efficient algorithm in the literature for constructing the MST, and use it with A graph search to solve TSP:*

Implemented using: Prim's MST

Link to github Repo (with demo): GITHUB

c) Study and report current state of the art methods which show (significant) improvement over MST heuristic:

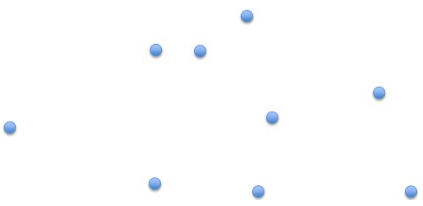
Christofides Algorithm

This algorithm is a well-known approximation algorithm for the metric traveling salesman problem. It assures that the solution for TSP will be in 1.5 factor of the optimal solution length.

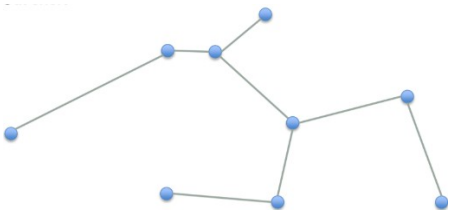
Let us consider a graph B which is complete and is defined on V, the set of vertices and function w which allocate a value greater than equal to zero to every edge of graph. B= (V, w), an example of TSP.

Steps of Christofides algorithm: -

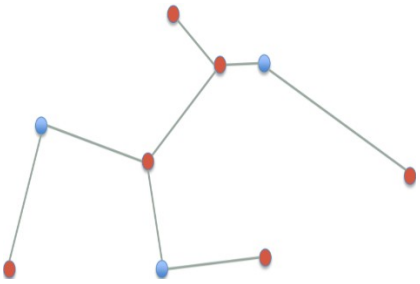
- Construct a minimum spanning tree S from B.
- Now, consider D, which is the set of odd degree vertices. The count of vertices in this set is even.
- Find a minimal matching R of the vertices in D.
- Add the edges of R and S and construct a connected multigraph Q. All the vertices in Q are of even degree.
- Find a Euler tour in Q.
- Short cut, visit the nodes as Euler tour but, skip the visited nodes



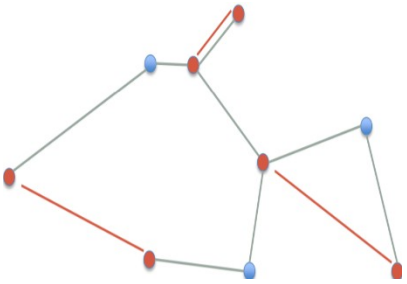
a)



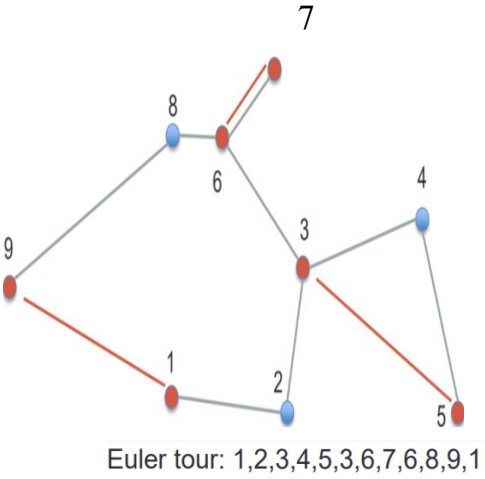
b)



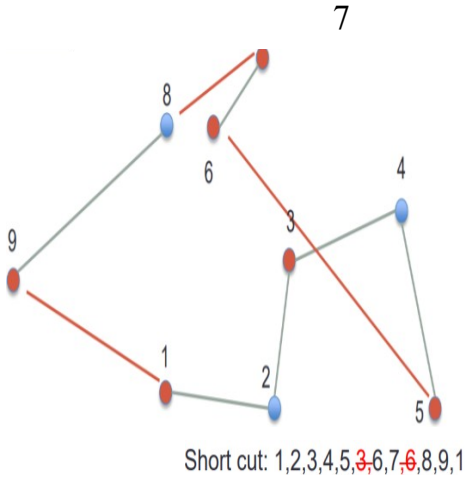
c)



d)



e)



f)

Figure: - Christofides Algorithm explained with the help of example.

It solves the TSP within 3/2 of the optimum. This can be assured by assuming P be the optimal TSP.

Now we remove an edge from P, and a spanning tree is formed. The weight of this spanning tree must be greater than the weight of minimum spanning tree, so it implies $w(S) \leq w(P)$. Now, consider count of odd degree vertices (D) in cyclic manner over C and partition the C into 2 set of paths: -

- Set A: - In this set, first path vertex has an odd count in cyclic manner.
- Set B: - In this set, first path vertex has an even count in cyclic manner.

Each set maps to minimal matching of odd degree vertices (D) that matches the two endpoints of each path. The weight of matching is at most equal to the weight of the paths. As set A and set B has partitioned the edges of P, either set A or set B has less than or equal to half of the weight of P and with the help of triangle inequality, the weight of corresponding matching is

less than or equal to half of the weight of P. The minimal matching does not have greater weight, so $w(R) \leq w(P)/2$. The Euler tour weight is equal to the weight of R and S.

$$W(\text{Euler tour}) = w(R)+w(S) \leq w(P)/2 + w(P) \leq 3 * w(P)/2$$

By triangle inequality, we can say that short cutting does not add any weight, so the solution will always be in 3w(P)/2 for Traveling Salesman Problem.

Reference/Bibliography:

1 Google

2 Google Scholar

3 Researchgate

4 Medium.com

5 Stackoverflow.com

Contributions: -

***a*) Study and solve it using minimum-spanning-tree (MST) heuristic:** - Studied and solved by Abhay Singhal and Sameer Gupta.

b) Find and implement an efficient algorithm in the literature for constructing the MST, and use it with A* graph search to solve TSP: - Implemented by Hritwik Singhal

c) Study and report current state of the art methods which show (significant) improvement over MST heuristic: - Studied and solved by Vani Agarwal, assisted by Abhay Singhal and Sameer Gupta.

Project Report written by: Vani Agarwal

Link to github Repo (with demo): GITHUB