Using Christofides Algorithm

Mohammed Al-Khrashi Mohammed Al-Ohaydab

438103043 438101307

1. Introduction

The Traveling Salesman Problem (TSP) is one the most famous computer science problems that has been studied intensively. The problem description is as follows: A salesman must travel between N cities cities = $\{c_1, c_2 .., c_n\}$, such that he visits every city exactly once, and returns back to the original city. Between every two cities c_i , c_j there is an associated cost $w_{i\,j}$, the salesman wants to know the path with the least cost. We will use Christofides Algorithm to find an approximate Solution to the TSP.

2. Christofides' Algorithm

Christofides' Algorithm is an alternative algorithm for solving the TSP, the algorithm is an approximation algorithm for finding the solution to the traveling salesman problem (TSP). The total cost that is produced by this algorithm is guaranteed to not exceed 1.5 of the optimal solution cost. The algorithm assumes an input graph G, where G is a complete graph (fully connected graph), and assumes that G is a metric graph, that is the cost (weight) between each two vertices in G satisfies the triangle inequality.

The algorithms steps can be written as follows:

1. Find a minimum spanning tree (MST) T for G
2. Let O be the set of vertices with odd degrees in T.
3. Find the minimum weight perfect matching M for the vertices in O.
4. Combine M and T to form a multigraph H
5. Form an Eulerian circuit in H
6. Make the circuit Hamiltonian by skipping every repeated vertex. This is our path.

3. Algorithm Steps Description and Implementation

In this section we will describe the algorithm steps, and how we decided to implement them.

A. Data Structures

To represent our graph G, we store our vertices in a Hash Table, where the vertex name is it's key, and it's value is a pair is it's euclidean coordinates, this is done to preserve the metric graph property.

We store our edges in a Hash Set

.....

B. Finding the MST

To find an MST for our graph G we used Kruskal's algorithm.

Kruskal's Algorithm is a greedy algorithm for finding an MST, it does that by adding edges to a new graph T from G based on the weights (increasingly), if adding a certain edge creates a cycle discard it, otherwise add it to the new graph T.

Pseudocode of Kruskal's Algorithm:

...

...

..

C. Odd degree Vertices

To find the odd degree vertices, simply iterate over all vertices in T (MST for G), and check if each vertex has an odd number of neighbors, if so add it to our list O. There should be an even number of vertices by the handshaking lemma.

D. Minimum Weight Perfect Matching of the Odd Vertices

A perfect matching of a graph is where each vertex of a graph is connected to exactly one other vertex.

To find the minimum perfect matching M for the odd vertices, we use a greedy approach that takes the first vertex in O, finds the vertex closest to it then creates an edge between them in M and removes both vertices from O, until O is empty.

Note that since we have an even number of odd degrees, all vertices will have exactly one edge.

...

...

..

E. Forming the Multigraph

Using M and T we form a multigraph H, note that a multigraph allows repeated edges.

F. Eulerian Tour

From our multigraph H, we start from our root vertex V (the vertex where we start from and return to).

We choose any neighbor of V and move to it, deleting the edge between them, if a vertex has no neighbors add it to the path, and go to the previous vertex (using a stack).

...

..

.

G. Hamiltonian Tour

Finding our final path is simple.

Move along the Eulerian circuit, skipping each repeated vertex. This path will now be our TSP solution.

4. Experiment and Data Interpretation

....

..

.

5. Conclusions

When finding an optimal solution for a TSP is unfeasible for large input size due to huge time complexity, approximation algorithms can be a valid alternative.

Christofides Algorithm provides us with a good approximation assuming that the graph is metric and complete.

The algorithm combines finding the MST of a graph, and finding a Minimum Weight Perfect Matching of a graph to find the final path, with these two algorithms having polynomial running time.

Christofides Algorithm finds a path in polynomial time. A complexity that is way more feasible than standard algorithms for finding a TSP. With a path cost that cannot exceed 3/2 the cost of the optimal solution.