# The Travelling Salesman Problem Variation in a maze

Ishita Chaudhary 2019CS10360

Swaraj Gawande 2019CS10406

27 July 2021

# 1 Introduction

The traveling salesman problem is a very famous and simple to understand problem but is very difficult to solve and for bigger sizes of parameters is practically impossible. It is an NP-hard problem, i.e., it cannot be solved precisely with time complexity less than exponential. Here in this report, we try to analyze the problem and its possible solutions. Firstly to analyze the scale of the problem, we discuss the conventional approach to solve TSP.We further specify some constraints and specify the exact problem we want to discuss.

# 2 Travelling Salesman Problem

## 2.1 Modelling the problem

The statement of the problem goes "Given a list of cities and the distances between each pair of cities, what is the shortest possible route that visits each city exactly once and returns to the origin city?". As it is important in theoretical computer science and operations research a lot of efforts have been put to heuristically solve the problem. But before understanding the solutions first we need to construct a mathematical model of the problem. Graph serves the purpose. The cities are the nodes of the graph and the distances between each pair of cities is the weight of that corresponding edge. Example let G be a completely connected graph (V,E) where is V is the set of vertices and E be the set of edges and their corresponding weights(no negative), S is the starting vertex. Then Travelling Salesman Problem models to finding a closed walk passing through all the vertices of minimum sum of weights of edges involved i.e. as the graph is complete there is at least one Hamiltonian cycle in the graph and we need find the minimum weight Hamiltonian cycle.

## 2.2 Possible Solutions

### 2.2.1 Brute force approach

A simple solution for TSP is to consider all the possible closed walks passing through all the vertices exactly once i.e. considering all the possible Hamiltonian cycles and then choose the one with minimum weight.

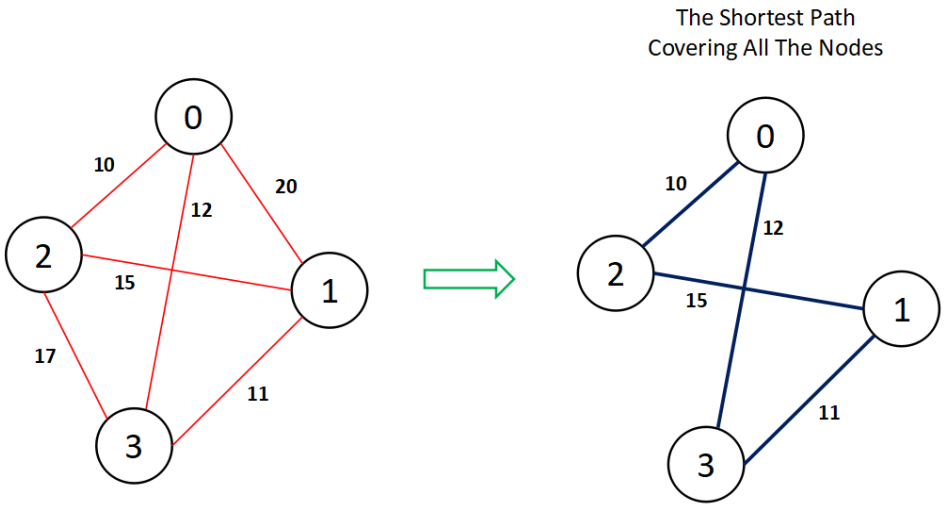


Figure 1: Hamiltonian Cycle for Weighted Graph

**Algorithm, Complexity and heuristic analysis**

```
Start from node S.
Generate all ($|V$-1)! Permutations of remaining nodes.
Calculate cost of every permutation and keep track of minimum cost permutatio
Return the permutation with minimum cost.
```

Starting with node S to get a Hamiltonian cycle we have $|V|-1$ options of edges to connect the second node and $|V|-2$for the second edge and so on and then connecting it back to S. So there are a total of $(|V|-1)!/2$ Hamiltonian cycles present in a complete graph with V as set as vertices as each cycle is counted twice. Now it would take $O(|V|)$ time to calculate the weights of each cycle and maintain the minimum thus a total of $O((|V|)!)$ time complexity for the complete algorithm. Even with about 15-20 nodes this task is just impossible for the fastest computers in reasonable time.
This approach gives the exact solution but it is optimal if time complexity is concern.

### 2.2.2 Using Dynamic Programming

In this approach we start with our starting node S and find the minimum weighted path from a vertex to any other vertex P and then the length of Hamiltonian cycle is C(S,P)+w(P,S) where C(S,P) is the minimum cost possible of a path from S to P containing all other vertices and w(P,S) is the weight of the edge connecting node P and S. Now to find C(S,P) we introduce Cs(S,P,$g$) which is the minimum cost of a path from node S to node P passing through all the vertices of connected sub-graph $g$ containing at least S and P. We start with $g$ containing only vertex S and P and then adding vertex one by one keeping track of the minimum path.

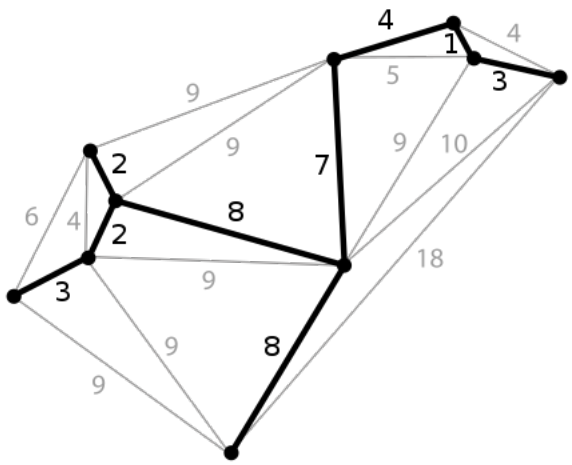


Figure 2: Minimum Spanning Tree of a Graph

Following is the pseudo code to calculate Cs(S,P,$g$)

```
If size of g is 2, then g must be {S, P},
 Cs(S,P,g) = w(S, P)
Else if size of g is greater than 2.
 Cs(S, P,g) = min { Cs(S,j,g-{P}) + w(j, P)} where j
 belongs to g, j != P and j != S.
```

**Complexity and heuristic analysis**
For a connected sub-graph with n vertices we need to calculate and store Cs(S,P,g) for each possible g. There are O($2^n$) possible g for each P and with maintaining minimum time complexity is O($n * 2^n$). There are O(n) possible Ps thus the overall time complexity is O($n^2 * 2^n$). The above solution gives exact answer but exponential time complexity although less than factorial is not enough for relevant size of graphs.

### 2.2.3 Minimum Spanning Tree TSP

As exactly calculating the solution is not possible sometimes we may need an approximate solution. MST TSP thus provides us with a reasonable approximate solution in polynomial time complexity by using concepts of minimum spanning tree and triangle inequality. Firstly construct a minimum spanning tree with vertex S as its root using the famous Kruskal's algorithm or Prim's algorithm. And then we perform a pre-order traversal on the MST considering edge between two consecutive vertices which are not repeated as shown in the figure below. and then connecting the last vertex to S.

Correctness
It gives us a range of cost in which the minimum cost of Hamiltonian cycles lies although. As by definition MST has the minimum cost of all spanning trees i.e. say OS is optimal solution and MST is the cost of minimum spanning tree. The cycle giving optimal solution OS is just the cost of a linear spanning Tree and an edge joining first and the last vertex

which is less than or equal to cost of MST. And the cost of output of the algorithm say OT is not more than cost of complete eulerian walk on MST. Cost of a complete walk of MST say CW costs no more than twice the cost of MST then

$$MST \leq OS \tag{1}$$

$$CW \leq 2 * MST \tag{2}$$

$$OT \leq CW \tag{3}$$

Also as a complete walk of the MST is also a cycle passing through all the vertices thus the optimal solution OS is never more than cost of complete walk CW which in turn is less than the twice of cost of MST by equation (1) we have

$$MST \leq OS \leq OT \leq 2 * MST \tag{4}$$

**Complexity Analysis** Finding Minimum Spanning Tree takes O($|V|^2$) time and then we perform a pre-order traversal and then one pass eliminating the repeated vertices which takes O($|V|$) time so over all time complexity of the algorithm is O($|V|^2$).

# 3 The Lazy Mario

## 3.1 Problem Statement

We want to introduce the lazy Mario, the protagonist of our story. As the name suggests, our prime character is extremely slothful and starving at the same time. Mario has a map of New Donk City, with all supermarket places marked on it.
Now, Mario must decide if he wants to step out in the scorching heat of the summers to buy all the essentials he will need to make his lunch or order the same on Zomato.
Mathematically speaking, the dissatisfaction coefficient would be in terms of the length of the path he needs to cover or the money he would have to spend.
The constraints being:

1. If he wishes to order on Zomato, he has a fixed sum of money he can spend say this gives him a dissatisfaction worth $z$ units.

2.If he chooses to go out, he will be frustrated if he has to travel more and switching a cell in the maze would give him dissatisfaction worth 1 unit.

It is interesting to note that Mario need not visit every supermarket in the city; he would minimize his visit to those places which satisfy his list of ingredients, i.e., the Steiner nodes of our problem. If he steps out, he must visit all Steiner nodes and return home to have his lunch.

We will help Mario decide if he should really go to the supermarket(s) or should instead order on Zomato.
Now with the experience of solving TSP we try to figure out what is to be done in our problem. One thing is for sure we will not look for the cycle with minimum cost among all possible cycles as this is not possible for our computers on a practical size of maze nor is the dynamic programming solution useful here. We try to look for a heuristic solution.

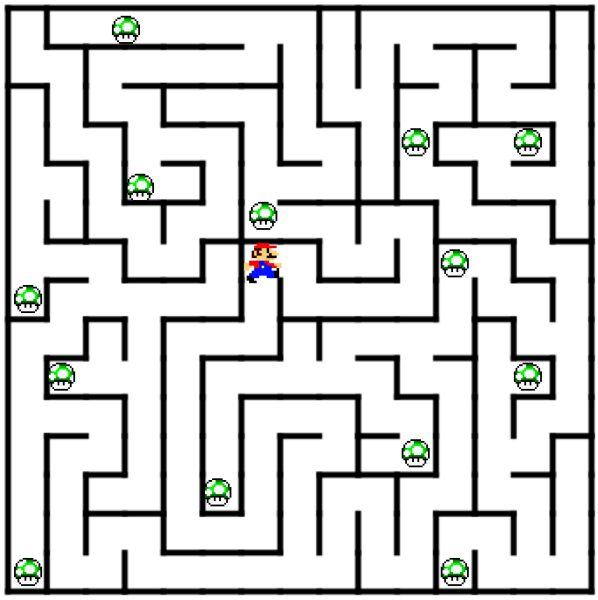

Figure 3: Map of the City with Mario's Home and Supermarkets Location

### 3.1.1 Case:Number of Steiner Nodes is very less

When number of Steiner nodes i.e. number of vertices we need to visit say n is very less than number of vertices in the graph $|V|$ for example $|V|$ 1000 and n¡10. We can perform n Breadth First Searches from each Steiner node and store cost of shortest path to each Steiner node and to starting node S which would take O($n^2$) space, and now we are left with a complete graph of n vertices and now we can use brute force approach or dynamic programming approach. So in this case we can directly

**Complexity and Heuristic analysis**

Doing BFS on n vertices is O(n($|V|+|E|$))and then further if used brute force approach then a total time complexity of O(n($|V|+|E|$)+n!) and space complexity of O($n^2$). if used Dynamic programming approach time complexity is O(n($|V|+|E|$)+$n^2 * 2^n$) and exponential space complexity in n. Also the graph we are considering is a maze which means no vertex has more than 4 edges thus $|E|$ is O($|V|$) so overall time complexity of BFS is O($n * |V|$).

### 3.1.2 Approximate solution using MST

Lets prepare our maze, although maze can be as it is treated as a graph but we need to first convert it into weighted graph. Converting a maze into a weighted graph involves starting with the graphical implementation of maze and visiting each node of the graph if the node is not a Steiner node and has exactly two neighbours then we delete that node and connect its two neighbours with an edge of weight equal to sum of the weights of the two edges incident to that node, as shown in the figure(Yellow nodes are Steiner nodes, green are terminal or branching nodes and red is starting node).

Now we can find a minimum spanning tree for this graph using famous Prim's or Kruskal's algorithm we cannot further use triangle inequality in this case there may not be any direct smaller path from a node to next node in the tree as in TSP as we now don't have complete

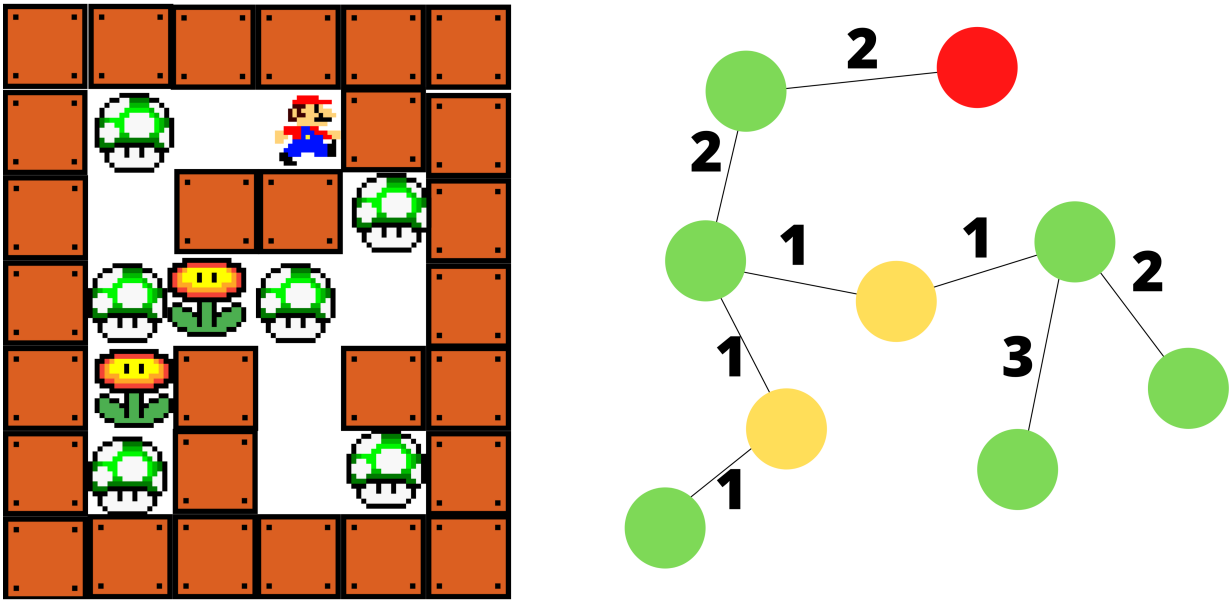


Figure 4: Creating a graph from the Maze
The flower in the map describes the Steiner Nodes, denoted by yellow nodes in the graph. The red node denotes the location of Mario's home. The edge weight depicts the length of the path.

graph. But the argument that the minimum cost of visiting each Steiner nodes is no more than cost of complete traversal on the MST. So in case Mario asks for our suggestion and the cost of MST is $m$ then if case 1

$$z > 2 * m \tag{5}$$

then we would suggest him to go out and travel doing a pre-order traversal of the MST. Although we cannot say that this would result in least dissatisfaction possible but it sure is better than home delivery. case 2

$$z < m \tag{6}$$

then we can clearly suggest him to go for home delivery as travelling to each supermarket is for sure more dissatisfying although we did not found the exact min cost path but min cost of visiting each supermarket is no less than cost of MST. case 3

$$m < z < 2^{*}m \tag{7}$$

and if the number of nodes in graph or the number of Steiner nodes is not what fits into the ranges given in the previous solution then we cant actually help Mario in this matter as the solution using MST is a heuristic one. But what we know here for sure whatever Mario decides he could not be at more loss than $m$ units.

**Complexity analysis** Converting a maze into a weighted graph takes O($|V|$).As discussed in TSP solutions finding MST takes O($n^2$) time and MST traversals are O(n). So the overall time complexity of the solution is O($|V| + n^2$).