Exploring the Traveling Salesman Problem

Luke Antonov

August 2024

1 Introduction

1.1 Abstract

Christofides's Algorithm is an algorithm related to the symmetric Traveling Salesman Problem (sTSP). It approximates the solution within $\frac{3}{2}$ of the minimum Hamiltonian cycle. The algorithm takes a graph G which is assumed to be complete, weighted, undirected, and satisfies the triangle inequality. A simplified version of Christofides's Algorithm is the Double Tree Algorithm. Here, a visual representation of these two algorithms is provided.

1.2 Assumptions

As stated above, there are four assumptions that are required for the application of these algorithms. These constitute a metric TSP.

- Completeness
- Weights
- Indirection
- Triangle inequality is satisfied

Most importantly, the triangle inequality means that for any 3 vertices on graph G (x, y, and z):

$$d(x, y) + d(y, z) \leq d(x, z)$$

1.3 Graph

Below is the graph that will be used for the remainder of the report. It satisfies all the characteristics required by a metric TSP.

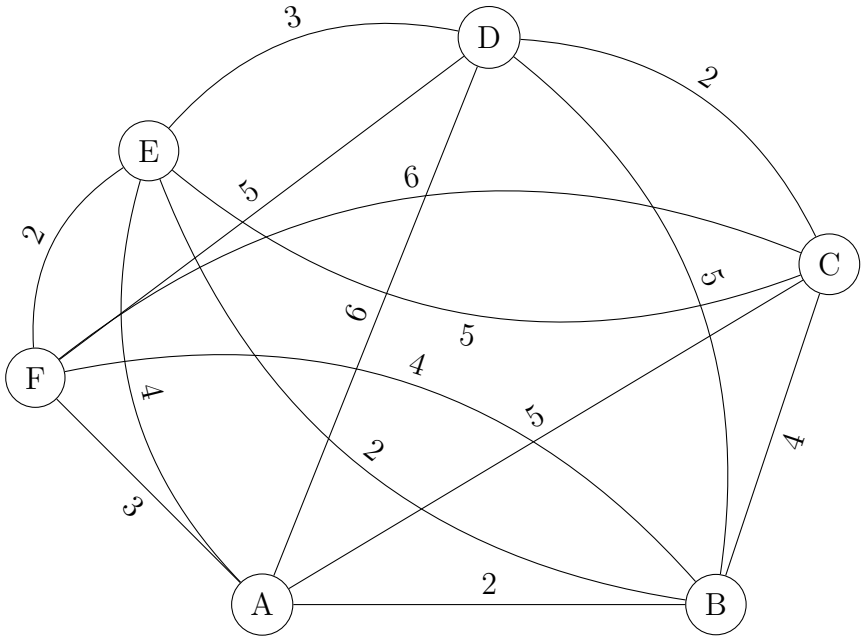


Figure 1: A weighted, undirected, complete graph with 6 vertices, which satisfies the triangle inequality.

2 Double Tree Algorithm

2.1 Introduction

The Double Tree algorithm allows for an approximation that is no more than 2 times the minimum Hamiltonian cycle.

2.2 Double Tree (Eulerian graph H)

The minimum spanning tree T^* is found using Kruskal's algorithm, which is discussed further below and is skipped here. For this method, we employ a DFS traversal to obtain a double tree of the minimum spanning tree (MST).

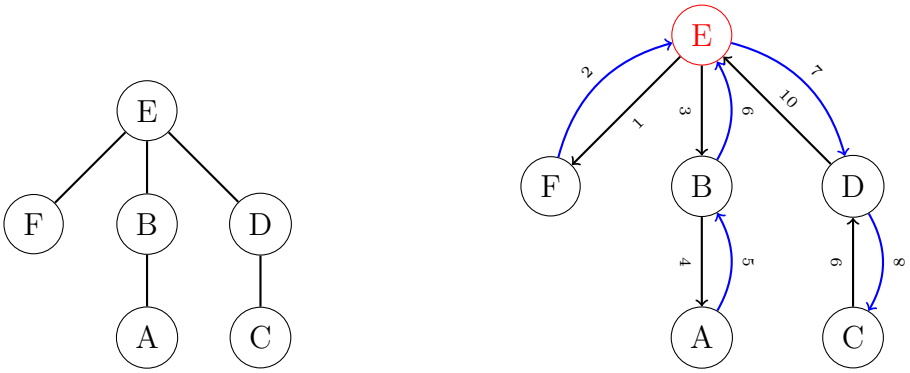


Figure 2: Deriving Eulerian graph H from MST. The numbers on the right figure are NOT edge weights, they are the edges enumerated in order of the Eulerian Tour W.

Here, the MST is converted into a double tree by employing a depth-first search (DFS) traversal. This creates a double tree, which allows for a Eulerian tour of the MST. A DFS works by starting at the root and then traversing as far down the left side as possible, then backtracking to the previous fork and continuing down. In this case, the traversal can be generalized as:

$$E \to F \to E \to A \to E \to C \to E$$

2.3 Hamiltonian Cycle

To construct a Hamiltonian cycle, first an Eulerian tour W is constructed from the double tree.

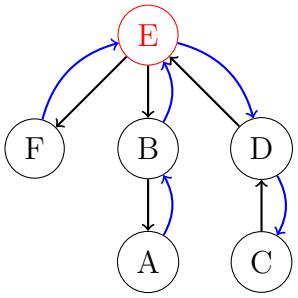


Figure 3: Eulerian graph H

The Eulerian tour of H is the path originating from the root and traversing all the vertices. In this case, it will be:

$$E \to F \to E \to B \to A \to B \to E \to D \to C \to D \to E$$

Now, every duplicate vertex other than the starting and ending ones can be removed. This is only possible because graph G is complete and adheres to the triangle inequality. Taking the first duplicate of vertex E:

$$F \to E \to B$$

Firstly, $F \to B$ exists as graph G is complete, meaning for any two vertices x and y, edge xy exists. Meaning we can draw:

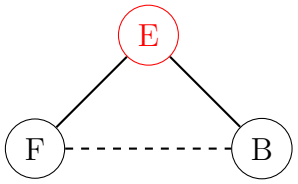


Figure 4: Triangle formed by vertices F, E, and B

Continuing, given the triangle inequality:

$$d(F, E) + d(E, B) \geq d(F, B)$$

we can conclude that E can be bypassed because doing so is at an equal or lesser cost.

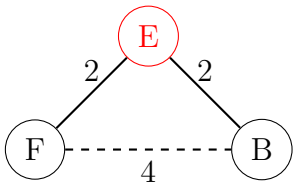


Figure 5: Triangle with weights included

Meaning from the Eulerian Tour of H:

$$E \to F \to E \to B \to A \to B \to E \to D \to C \to D \to E$$

a Hamiltonian cycle H_{dt} can be derived:

$$E \to F \to B \to A \to D \to C \to E$$

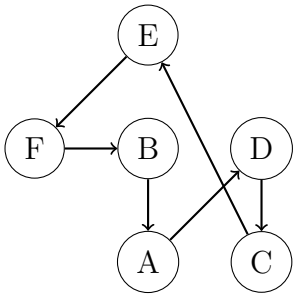


Figure 6: Hamiltonian cycle H_{dt}

2.4 Proximity to Optimal Value

Let our optimal minimum Hamiltonian cycle be H_G^* and W be the Eulerian tour of H. This means:

$$\mathrm{cost}(W) = \mathrm{cost}(2T^*)$$

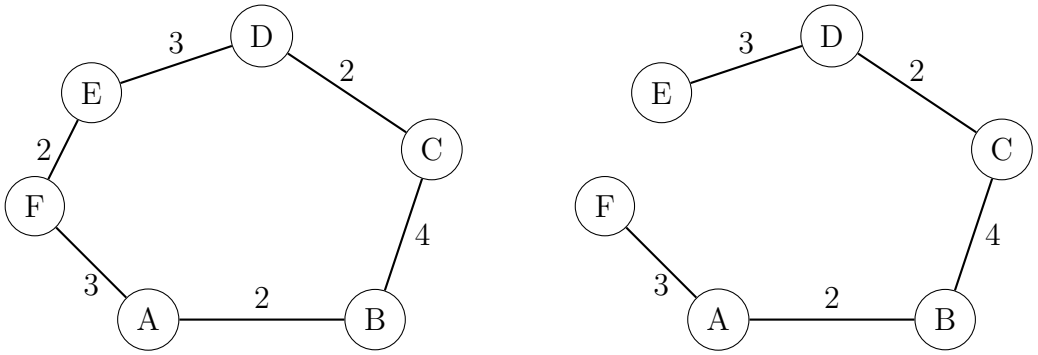
This is derived from Figure 7a. Edges 1 and 2 are equal, edges 3 and 6 are equal, and so on. Therefore, the cost of W is twice the cost of edges 1, 3, 4, 6, and 7. The cost of these edges is also equal to the cost of T^*, the minimum spanning tree.

$$\mathrm{cost}(W) = 2 \cdot \mathrm{cost}(\text{Edge 1, 3, 4, 6, 7}) = 2 \cdot \mathrm{cost}(T^*)$$

It can also be concluded that the cost of W is greater than or equal to the cost of our Hamiltonian cycle derived from the algorithm, H_{dt}, because each shortcut in the Hamiltonian cycle is equal to or less than its equivalent path in W due to the triangle inequality. Meaning:

$$\mathrm{cost}(H_{\mathrm{dt}}) \leq \mathrm{cost}(W)$$

$$\mathrm{cost}(H_{\mathrm{dt}}) \leq 2 \cdot \mathrm{cost}(T^*)$$



(a) H_G^* (b) Tree T_H

Figure 7: Deriving Tree T_H from ideal Hamiltonian Cycle H_G^*

Cost(T_H) must be greater than or equal to cost(T^*) by definition because T^* is the minimum spanning tree. Meaning

$$\text{cost}(T_H) \geq \text{cost}(T^*)$$

Since cost(T_H) is just cost(H_G^*) with an edge removed, we have cost(H_G^*) $\geq$ cost(T_H) $\geq$ cost(T^*). This means that

$$2\text{cost}(H_G^*) \geq 2\text{cost}(T^*)$$

Finally, meaning

$$\text{cost}(H_{\text{dt}}) \leq 2 \cdot \text{cost}(T^*) \leq 2 \cdot \text{cost}(H_G^*)$$

With this, the conclusion is reached that the Double Tree Algorithm is at worst twice the value of the ideal minimum Hamiltonian cycle H_G^*.

3 Christofides's Algorithm

Christofides's Algorithm is a well-known approximation algorithm for solving the symmetric Traveling Salesman Problem (sTSP). It guarantees a solution within $\frac{3}{2}$ times the optimal minimum Hamiltonian cycle.

3.1 Kruskal's Algorithm

To construct a minimum spanning tree T^* from graph G, Kruskal's Algorithm can be utilized.

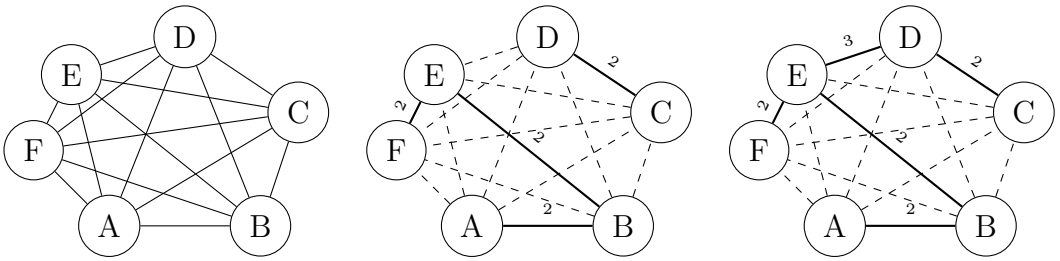


Figure 8: Steps to find the MST using Kruskal's algorithm.

Kruskal's Algorithm works by selecting edges of increasing weight until all vertices are reached while avoiding closed loops. From the initial graph, first, all the edges of the lowest weight (2) are selected. Since none of them form a closed loop, they are all valid for Kruskal's Algorithm. Then, only the E-D edge of weight 3 is selected, as the A-F edge, which is also of weight 3, would cause a closed loop.

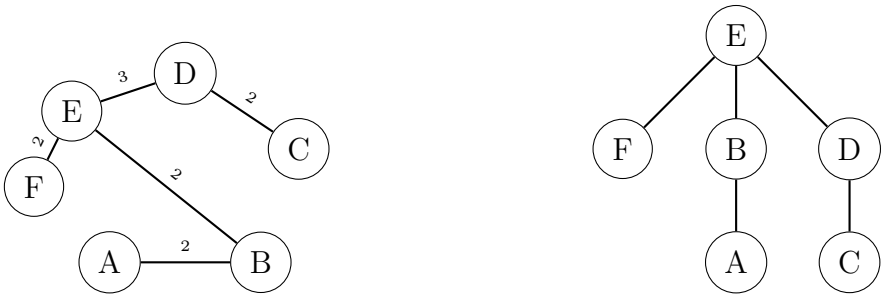


Figure 9: Choosing Vertex E as the root and redrawing.

Once a minimum spanning tree T^* is found, a node can be selected as the root of the tree, and it can be redrawn for easier readability.

3.2 Subgraph with Odd Vertices

After the MST is obtained, a subgraph O consisting of all vertices with odd degrees is created. These odd-degree vertices must be paired to form a perfect matching, ensuring that all vertices in the resulting graph have even degrees. This step is crucial for creating an Eulerian circuit in the next phase of Christofides's Algorithm.

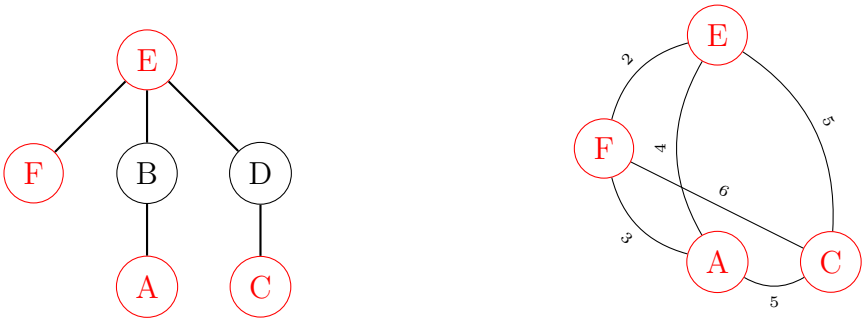


Figure 10: Deriving subgraph O from odd degree vertices in T^*.

Here, our odd vertices are F, E, A, and C. From these vertices, subgraph O is made.

3.3 Minimum Perfect Matching

Perfect Matching means that every single vertex is connected with one edge coming out of it. Minimum perfect matching is when this is done while achieving the minimum possible weight.

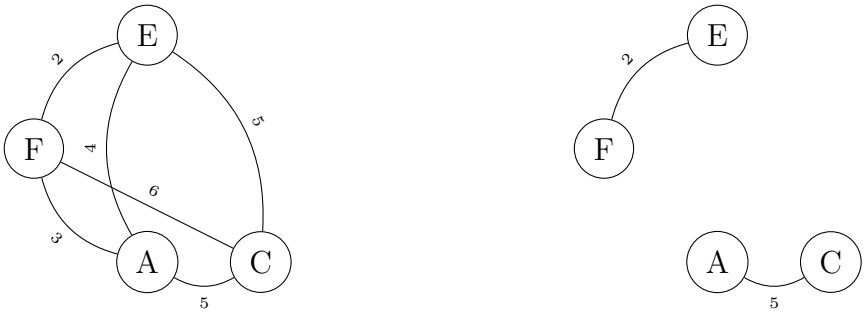


Figure 11: Finding minimum perfect matching M^* of subgraph O.

3.4 Combine M^* and T^*

Now, both the minimum spanning tree T^* and the minimum perfect matching M^* are combined to create an Eulerian graph H. Double edges are allowed. This step is similar to the Double Tree Algorithm, but this time we are more selective with our double edges.

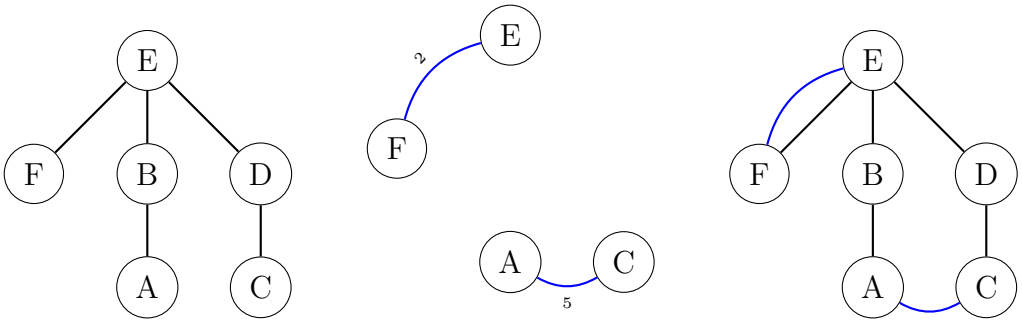


Figure 12: Finding minimum perfect matching of subgraph O.

3.5 Construct Eulerian Tour

Just like in the Double Tree Algorithm, we construct a tour to hit all the vertices.

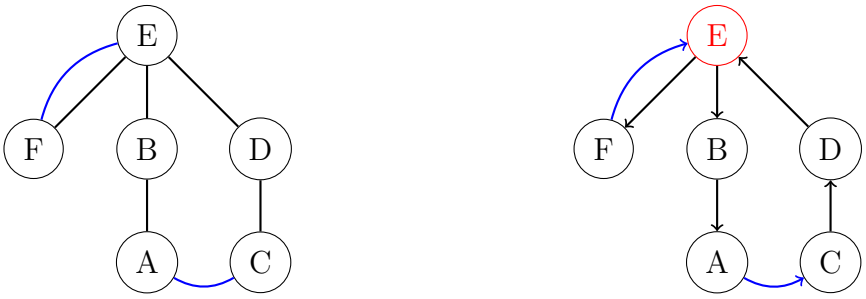


Figure 13: Creating Eulerian Tour W from Eulerian Graph H

Here, the Eulerian Tour starting at the root node E is given by the following:

$$E \to F \to E \to B \to A \to C \to D \to E$$

3.6 Hamiltonian Cycle

We only want to visit each vertex once, so in this case duplicate vertices can be removed like in the Double Tree Algorithm:

$$\text{i. } E \to F \to E \to B \to A \to C \to D \to E$$

$$\text{ii. } E \to F \to B \to A \to C \to D \to E$$

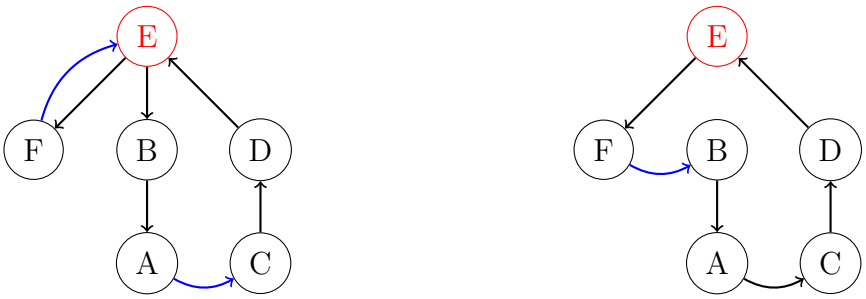


Figure 14: Finding Hamiltonian Cycle H_s from Eulerian Tour W

4 Final Calculation

Now, the calculations of the ideal minimum Hamiltonian cycle H_G^*, the Double Tree Algorithm H_{dt}, and the Christofides Algorithm H_s will be compared.

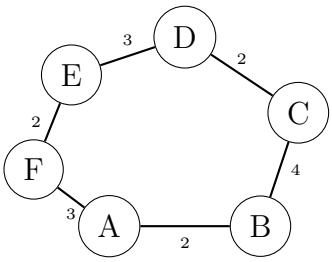


Figure 15: H_G^*

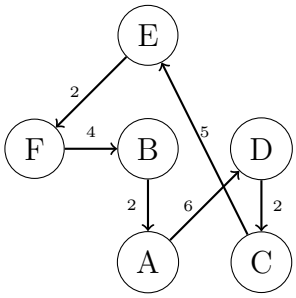


Figure 16: H_{dt}

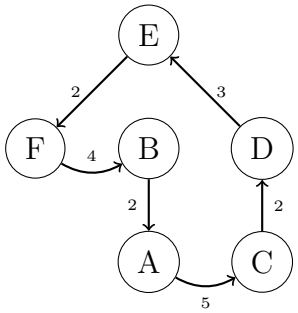


Figure 17: H_s

Figure 18: Comparison of Cost

Adding up the final costs for each of the Hamiltonians, we get $\text{cost}(H_G^*) = 16$, $\text{cost}(H_{\text{dt}}) = 21$, $\text{cost}(H_s) = 18$. This means that $\text{cost}(H_G^*) < \text{cost}(H_s) < \text{cost}(H_{\text{dt}})$ and:

$$\text{cost}(H_s) < \frac{3}{2} \cdot \text{cost}(H_G^*)$$

$$\text{cost}(H_{\text{dt}}) < 2 \cdot \text{cost}(H_G^*)$$

Which is what is expected from both the Double Tree and Christofides Algorithms.

5 Sources

References

[1] Richard, E. (2024). *Graph Theory*. Nagoya University. Retrieved from https://www.math.nagoya-u.ac.jp/~richard/teaching/s2024/Graph.pdf

[2] MIT OpenCourseWare. *R9. Approximation Algorithms: Traveling Salesman Problem* [Video]. YouTube. Retrieved from https://www.youtube.com/watch?v=1FEP_sNb62k

[3] EducateYourself. *Minimum Spanning Tree* [Video]. YouTube. Retrieved from https://www.youtube.com/watch?v=Yo7sddEVONg

[4] Leiserson, C., Rivest, R., Stein, C., & Cormen, T. (2015). *Design and Analysis of Algorithms*. MIT OpenCourseWare. Retrieved from https://ocw.mit.edu/courses/6-046j-design-and-analysis-of-algorithms-spring-2015/