2-Approximation Algorithm for the Minimum Weighted Steiner Tree Problem

Madhumitha Thota
CSE
National Institute of Technology Warangal
tm21csb0a61@student.nitw.ac.in

J Bhavishya
CSE
National Institute of Technology Warangal
jb21csb0a24@student.nitw.ac.in

Tanu Priya
CSE
National Institute of Technology Warangal
tp21csb0a58@student.nitw.ac.in

I. Introduction

The report focuses on the application of approximation algorithm in solving the Steiner Minimal Tree Problem (SMT). Traditional approaches like Exact Algorithm and Heuristic Algorithms are explored, highlighting their limitations and strengths. The 2-Approximation Algorithm proposed here aims to provide a solution with a cost no more than twice the optimal solution cost. Furthermore, empirical evaluations are conducted to assess the performance of the algorithm on various instances of the Minimum Weighted Steiner Tree Problem (MWSTP). Subsequently, through C++ implementation, we analyze the theoretical guarantees of the algorithm, demonstrating its approximation ratio and time complexity.

II. Problem Statement

The Steiner tree problem in graphs can be seen as a generalization of two other famous combinatorial optimization problems: the (non-negative) shortest path problem and the minimum spanning tree problem. In the Minimum Weighted Steiner Tree Problem, we are given an undirected graph G = (V, E), where V represents the set of vertices and E represents the set of edges. Additionally, a subset R ⊆V of required vertices is specified, and each edge e in E is associated with a weight w(e). Our aim is to find a tree T that connects all the required vertices in R with the minimum total weight. Formally, the problem aims to minimize the sum of weights of the edges in the tree T, subject to the constraint that all vertices in R must be included in the tree.

III. Previous Approaches

The Minimum Weighted Steiner Tree Problem (MWSTP) has been extensively studied in the field of combinatorial optimization. Various algorithms have been developed to tackle this NP-hard problem, ranging from exact algorithms to approximation algorithms. In this section, we will review some of the key approaches to solving the MWSTP.

Exact Algorithms: Dreyfus-Wagner Algorithm

The Dreyfus-Wagner algorithm, devised by Dreyfus and Wagner, is a dynamic programming approach aimed at solving the Minimum Weighted Steiner Tree Problem (MWSTP). This algorithm iteratively constructs the Steiner tree by recursively computing the minimum total edge cost for each subset of terminals.

At its core, the algorithm employs dynamic programming to iteratively build the Steiner tree from the bottom up, starting with individual terminals and gradually expanding to larger subsets. It efficiently computes the minimum total edge cost by considering all possible combinations of terminals and intermediate vertices.

Input: $G = (V, E, w)$, $Y \subseteq V$;
Output: A minimum total edge cost of a subtree $T = (V', E')$, $Y \in V'$;

Compute P_{vw} for all $w, v \in V$;
for $(i = 2, \ldots, k-1)$ **do**
  For any $|X| = i$ and any $w \in Y$, $v \in V$:
    $T(X \cup Y) = min\{P_{vw} \cup T(X' \cup w) \cup T(X'' \cup w)\}$
end

Algorithm 1: Dreyfus-Wagner

Despite its effectiveness in providing exact solutions, the Dreyfus-Wagner algorithm's time complexity is exponential, specifically O(n^3 + n * 3^k), where n represents the number of vertices in the graph and k is the number of terminals. This exponential complexity limits its practical applicability to small or moderately sized instances of the MWSTP.

Heuristic Algorithms:

Heuristic algorithms offer approximate solutions to the Minimum Weighted Steiner Tree Problem (MWSTP) by prioritizing certain edges or vertices based on predefined criteria. Common heuristic strategies include greedy algorithms, local search algorithms, and genetic algorithms. While heuristic algorithms may sacrifice solution quality for computational speed, they often outperform exact algorithms in terms of efficiency.

Approximation Algorithms:

Approximation algorithms serve as a middle ground between exact and heuristic approaches by providing solutions guaranteed to be within a certain factor of the optimal solution. Particularly valuable for NP-hard problems where finding the optimal solution is computationally prohibitive, approximation algorithms offer a practical compromise. For example, the 2-Approximation Algorithm for the MWSTP ensures a solution within a factor of 2 of the optimal solution. This algorithm constructs a Steiner tree that spans a subset of the input vertices while maintaining a certain level of connectivity.

IV. Proposed Algorithm

A 2-Approximation algorithm is proposed in which MST is used to approximate SMT. Given the original graph G = (V, E, w) and the set of terminal nodes $Y \subseteq V$.

To approximate the Minimum Weighted Steiner Tree Problem, the algorithm begins by constructing a metric closure on the set of terminal nodes Y of the original graph G. This metric closure forms a complete graph GY, where each edge's weight equals the length of the corresponding shortest path in G between terminal nodes using Djikstra's Agorithm. Subsequently, the algorithm computes a minimum spanning tree TY on the metric closure graph GY, using Prim's algorithm. Then, the minimum spanning tree TY is transformed into a Steiner tree by merging all shortest paths between pairs of terminals connected by an edge in TY. During this process, each edge e = (u, v) in the edges of TY is examined, and the shortest path P from vertex u to vertex v in G is determined. If P contains fewer than two vertices already in the Steiner tree T, the entire path P is added to T. Otherwise, sub-paths from vertex u to the first vertex already in T and from the last vertex in T to vertex v are added. Finally, any necessary post-processing steps are performed to remove possible cycles from the Steiner tree. The algorithm outputs the minimum total edge cost of the resulting subtree T, which encompasses all terminal nodes Y**.**

Assumption:
We have assumed that, in our graph edge costs satisfy the triangle inequality.

Input: $G = (V, E, w)$, $Y \in V$;
Output: a minimum total edge cost of a subtree $T = (V', E')$, $Y \in V'$;

Construct the metric closure G_Y on Y;
Find an MST T_Y on G_Y;
Set $T \leftarrow \emptyset$;
for *each edge* $e = (u, v) \subset E(T_Y)$ **do**
  Find a shortest path P from u to v on G
  if P *contains less than two vertices in* T **then**
    Add P to T
  end
  else
    Let p_i and p_j be the first and last vertices already in T
    Add subpaths from u to p_i and from p_j to v
  end
end

Algorithm 2: Approximation

Approximation Ratio:

We will evaluate the quality of the solution of this algorithm in theory. Let $G_Y = (Y, E(G_Y), \overline{w})$. Note that $w(T) \leq \overline{T_Y}$, since at most all of the shortest paths between each pair of terminals are inserted. We want to compare T_Y with an SMT. Let T^*_Y be the SMT. Consider a Eulerian tour X on the doubling tree. Since the tour passes each edge exactly twice, we have $w(X) = 2(T^*_Y)$. On the other hand, since the tour visits all the terminals in the graph, we have

$$w(X) \geq w(tsp(G_Y)) \geq w(T_Y),$$
$$w(T) \leq w(TY) \leq 2 \times w(T^*Y)$$

To see that the upper bound is asymptotically tight, consider an extreme example. Suppose that there are k terminals, and 1 nonterminal. The terminals form a cycle with w(e) = 2 for each edge e in the cycle. The nonterminal is adjacent to each terminal with an edge of length 1. The SMT is the star centered at the nonterminal in the middle of the graph and has total cost k. Meanwhile, an MST is the path consisting of the terminals and has total cost 2(k − 1). Therefore, the Steiner ratio is

$$\rho = w(MST(G_Y)) / w(SMT(G_Y)) = 2(k-1) / k = 2-2/k.$$

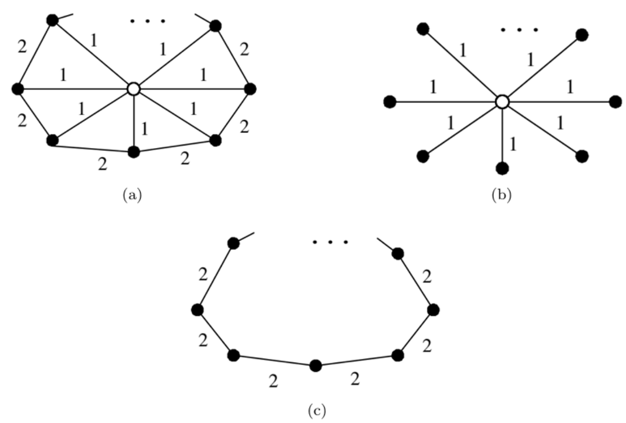

Figure 1: (a) A graph in which the white vertex is a nonterminal, and others are terminals. (b) An SMT. (c) An MST of the subgraph induced on the terminals.

Hence, the approximation is a 2-approximation. The Steiner ratio in general metric is 2, and is smaller in the Euclidean metric as the Euclidean space is a restricted version of the general metric.

Time Complexity:
The time complexity of the algorithm is O(V*K^2). The majority of the running time is due by the construction of the metric closure.

Djikstra's Algorithm: O((V + E) log V), where V is the number of nodes and E is the number of edges in the graph G.

Metric Closure: It takes O(K * (V + E) log V) time.

Prim's Algorithm: It runs in O(V^2) time.

Steiner Tree Construction: O(K^2) since there are K terminal nodes.

V. References

[1] Anh T. Tran and Tam T. Nguyen Econometrics and Operations Research, School of Business and Economics, Maastricht University. 2-Approximation Algorithm for the Minimum Weighted Steiner Tree Problem.

[2] A. B. Kahng and G. Robins, On optimal interconnections for VLSI (Springer Science and Business Media, City Address, (Vol. 301)).

[3] B. Y. Wu and K.-M. Chao, Spanning trees and optimization problems (CRC Press, 2004).

[4] A. Segev, The node-weighted steiner tree problem, Networks 17, 117 (1987).

VI. Results

Input1:

```
SECTION Graph
Nodes 8
Edges 21
E 1 2 2
E 1 3 2
E 2 3 2
E 2 4 1
E 2 5 1
E 3 4 1
E 3 5 1
E 4 5 1
E 4 6 2
E 5 6 2
E 1 4 3
E 1 5 3
E 2 6 3
E 3 6 3
E 4 6 4
E 4 7 3
E 5 7 3
E 6 7 3
E 4 8 4
E 5 8 4
E 6 8 4
END
SECTION Terminals
Terminals 4
T 1
T 2
T 3
T 4
END
EOF
```

Output1:

```
Metric Closure Matrix:
0 2 2 2 2 3 3 4
2 0 1 1 1 2 3 4
2 1 0 1 1 2 3 4
2 1 1 0 1 2 3 4
2 1 1 1 1 2 3 4
3 2 2 2 2 2 3 4
3 3 3 3 3 3 3 4
4 4 4 4 4 4 4 4
Weight of Minimum Spanning Tree (MST): 14
Steiner Tree weight: 9

--------------------------------
Process exited after 6.065 seconds with return value 0
Press any key to continue . . .
```

Input2:

```
SECTION Graph
Nodes 7
Edges 11
E 1 2 2
E 1 3 4
E 2 3 1
E 2 4 7
E 3 4 3
E 3 5 5
E 4 5 1
E 4 6 8
E 5 6 2
E 2 7 1
E 6 7 1
END
SECTION Terminals
Terminals 3
T 1
T 3
T 5
END
EOF
```

Output2:

```
Metric Closure Matrix:
0 2 3 3 4 3 3
2 1 1 3 4 3 2
3 1 0 3 4 3 2
3 3 3 1 1 2 3
4 4 4 1 0 2 3
3 3 3 2 2 2 3
3 2 2 3 3 3 2
Weight of Minimum Spanning Tree (MST): 11
Steiner Tree weight: 13

--------------------------------
Process exited after 5.966 seconds with return value 0
Press any key to continue . . .
```