The Steiner Tree Problem

Chirag Wadhwa: 14

Suraj Dev Deka: 51

Table of Contents

1. Overview

The Steiner Tree Problem (STP) is a classic NP-hard problem in graph theory and combinatorial optimization. It seeks the cheapest way to connect a required set of points, allowing for the use of extra "junction" points to reduce the total cost. Its applications are widespread, from designing electronic circuits to routing network traffic.

2. Problem Statement

Given an undirected graph G = (V, E) with a positive weight $w(e)$ for each edge e $\in$ E, and a subset of vertices T $\subseteq$ V called **terminals**, the goal is to find a tree S = (V′, E′) that is a subgraph of G and satisfies two conditions:

1. It connects all the terminals (T $\subseteq$ V′).
2. Its total weight, $\sum_{e \in E'} w(e)$, is minimized.

The vertices in V′\T are called **Steiner points**.

2.1. NP-Hardness

The Steiner Tree Problem is NP-hard. The decision version of the problem was first shown to be NP-complete by Karp [1]. Subsequently, Garey, Graham, and Johnson [2] provided a more detailed analysis of its computational complexity and proved NP-hardness for both graph and geometric variants of the problem.

3. Brute Force Algorithm (BF)

This exact algorithm enumerates all subsets of potential Steiner nodes, computes the MST for each subset combined with the terminals, and returns the tree with the minimum cost. It guarantees the optimal Steiner tree but is exponentially slow, making it feasible only for small graphs.

```
Algorithm BruteForceSteinerTree(G = (V, E, w), T)

Input:

    G = undirected weighted graph

    T = set of terminal vertices

Output:

    Minimum-cost Steiner Tree connecting all terminals


1. bestCost ← ∞

2. bestTree ← ∅

3. S ← V \ T                     // Possible Steiner nodes


4. for each subset X ⊆ S do

5.     V' ← T ∪ X

6.     G' ← induced subgraph of G on V'

7.     if G' is connected then

8.         MST ← MinimumSpanningTree(G')

9.         cost ← total weight of MST

10.        if cost < bestCost then

11.            bestCost ← cost

12.            bestTree ← MST

13.        end if

14.    end if

15. end for


16. return bestTree, bestCost
```

4. Approximation Algorithm (App.)

This algorithm is a greedy, nearest-terminal heuristic that iteratively connects the closest unconnected terminal to the growing tree. Takahashi and Matsuyama [3] proved that the resulting tree has a cost at most twice the optimal, making it a 2-approximation algorithm. It runs in polynomial time and is practical for medium to large graphs.

4.1. Approximation Factor

$$2 \cdot OPT$$

More precisely,

$$2 \cdot (1 - 1 / k) \cdot OPT$$

Here, k is the number of terminal vertices that must be connected in the graph.

5. Heuristic Algorithm (Heu.)

This heuristic constructs a low-cost Steiner tree by first computing the Minimum Spanning Tree (MST) of the entire graph and then iteratively pruning any non-terminal leaf nodes. The remaining tree connects all terminals efficiently, is fast to compute, and is practical for medium to large graphs, though it does not guarantee an optimal or bounded approximation.

6. Experimental Setup

To evaluate algorithm scalability and performance, we generated synthetic graphs of varying sizes, categorized as Small, Medium, and Large. The parameters for each category are defined in the table below.

Graph Size	**Vertices**	**Edges**	**Terminal**	**Instances**	**Algorithms Run**
Small	10-20	15-50	10–20%	5	BF, App, Heu
Medium	50-200	100-1000	10–20%	5	App, Heu
Large	500-2000	2000–10000	10–20%	5	App, Heu

6.1. Comparison 1: Comparison of App. and Heu. with Brute Force

For small graphs (10–20 vertices), the Takahashi–Matsuyama approximation and MST-pruning heuristic will be compared against the brute-force optimal solution. Metrics include tree cost, runtime, and Steiner nodes used.

The Brute Force algorithm became practically infeasible at **V=20**, where its runtime reached nearly 20 seconds, a 6-fold increase from the V=18 instance. This confirms its exponential **break point**.

Instance	Vertices	Edges	K	Brute Cost	Approx. Cost	Heuristic Cost	Approx/ Opt	Heurestic/ Opt
Test 1	17	40	3	147	162	147	1.102	1.000
Test 2	13	46	2	27	27	27	1.000	1.000
Test 3	18	42	3	153	153	196	1.000	1.281
Test 4	20	47	3	64	64	64	1.000	1.000
Test 5	11	33	2	56	56	73	1.000	1.303

For Small Graphs

6.2. Comparison 2: Comparison of App. and Heu.

For medium (50–200 vertices) and large (500–2000 vertices) graphs, only TM approximation and MST-pruning heuristic will be tested. Evaluation will focus on solution cost and runtime. TM approximation yields lower-cost trees, while MST-pruning is faster and simpler.

Instance	Vertices	Edges	K	Approx. Cost	Approx. Time(ms)	Heu. Cost	Heu. Time(ms)	Heuristic/ Approx
Test 1	60	658	7	73	3.331	86	2.442	1.178
Test 2	187	950	18	513	8.808	673	3.330	1.311
Test 3	103	379	20	548	4.928	612	1.606	1.116
Test 4	136	973	25	434	10.527	544	2.951	1.253
Test 5	73	755	7	111	2.185	132	2.902	1.189

For Medium Graphs

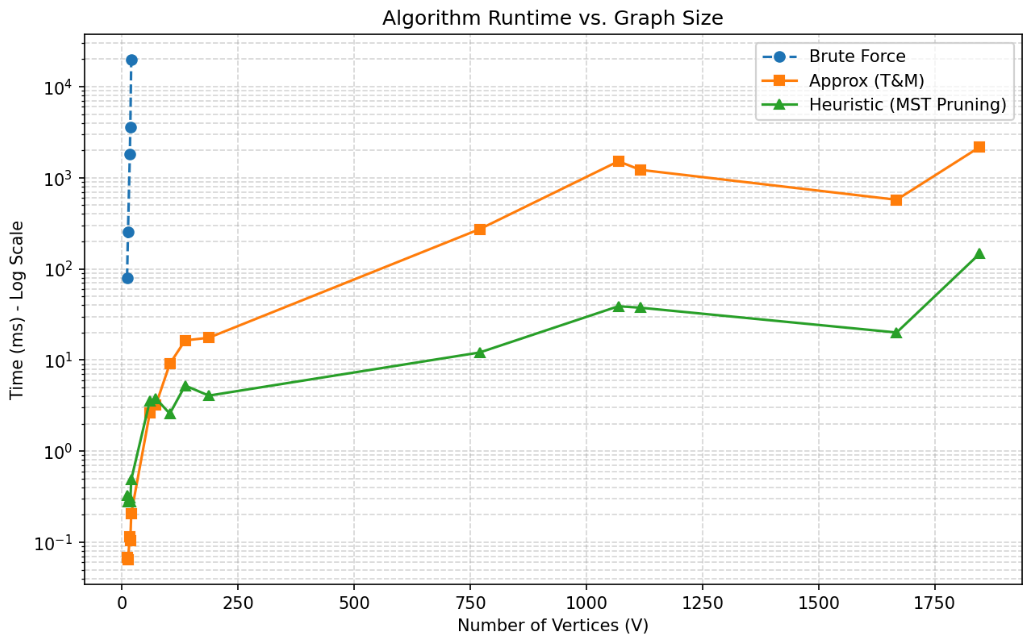
Instance	**Vertices**	**Edges**	**K**	**Approx. Cost**	**Approx. Time(ms)**	**Heu. Cost**	**Heu. Time(ms)**	**Heuristic/ Approx**
Test 1	1069	9158	192	2789	1189.612	3340	139.426	1.197
Test 2	1847	9300	240	5530	2131.1235	6815	43.906	1.232
Test 3	1668	2106	183	16586	592.425	19481	19.868	1.175
Test 4	770	2388	123	4888	245.584	5940	13.7142	1.215
Test 5	1116	9303	189	2758	1239.112	3288	35.2124	1.192

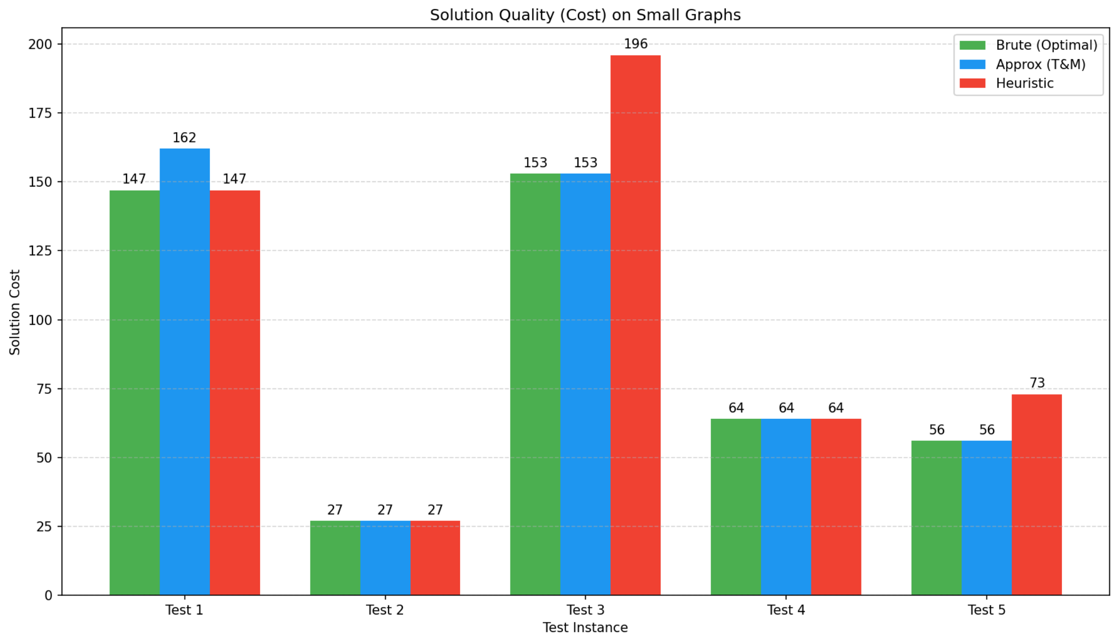
For Large Graphs

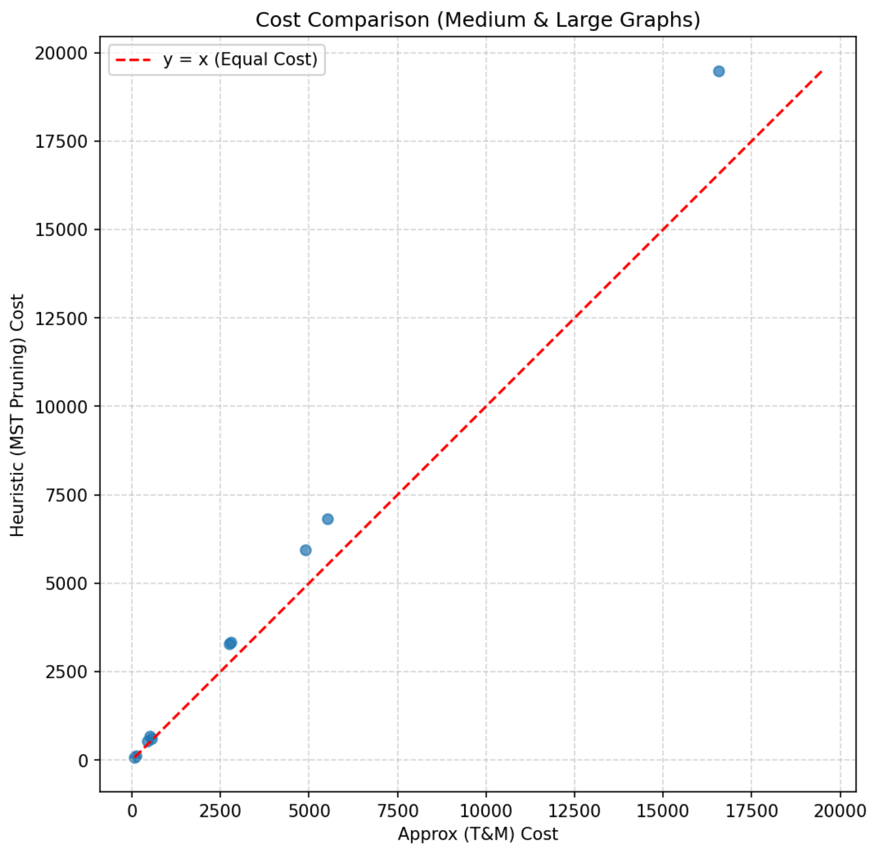
6.3. Results

1. Brute-Force (small graphs only):
    a. Provides the optimal Steiner tree, which used as a baseline to measure the quality of the other two algorithms
    b. Runtime grows exponentially exploding from 80 ms on an 11-vertex graph to 19,937 ms (nearly 20 seconds) on a 20-vertex graph, confirming it is infeasible for larger instances.
2. Takahashi–Matsuyama Approximation:
    a. Produces trees with costs very close to optimal. It found the perfectly optimal solution in 4 out of 5 small graphs.
    b. Runtime is highly efficient and scales well in polynomial time, solving large 1800-vertex graphs in 1-2 seconds.
    c. Consistently found better (cheaper) solutions than the MST-Pruning Heuristic across all medium and large test cases..
3. MST-Pruning Heuristic:
    a. Produces low-cost trees very quickly, but with no guarantee of optimality. In the small tests, it produced solutions that were up to 30% more expensive than the optimal (Test 5).
    b. It is the fastest algorithm by a large margin, solving even the largest graphs in just 37-147 ms.
    c. The solutions were consistently more expensive than the T&M approximation, in some large graph cases by 15-20%.
4. Overall Trends:

a. Small graphs**:** The T&M algorithm's superior quality was confirmed, as it found the optimal solution more frequently than the Heuristic.
b. Medium/Large graphs**:** A clear trade-off emerged: the MST-Pruning Heuristic is significantly faster, but the T&M Approximation is significantly more accurate (produces lower-cost trees).
c. The Brute-Force algorithm is clearly infeasible, while the T&M Approximation provides the best balance of high-quality solutions and efficient performance.







References

[1] R. M. Karp, *“Reducibility among combinatorial problems”, Complexity of Computer Computations*, R. E. Miller and J. W. Thatcher (Eds.), Springer, 1972, pp. 85–103. [Online]. Available: https://doi.org/10.1007/978-1-4684-2001-2_9

[2] M. R. Garey, R. L. Graham, and D. S. Johnson, *“The complexity of computing Steiner minimal trees”, SIAM Journal on Applied Mathematics*, vol. 32, no. 4, pp. 835–859, 1977. [Online]. Available: https://doi.org/10.1137/0132072

[3] H. Takahashi and A. Matsuyama, *“An approximate solution for the Steiner problem in graphs”*, *Mathematica Japonica*, vol. 24, no. 6, pp. 573–577, 1980. [Online]. Available: https://www.cs.haifa.ac.il/~golumbic/courses/seminar-2010approx/takahashi80.PDF