The Steiner Tree Problem

NP-HARDNESS, BRUTE FORCE, APPROXIMATION & HEURISTIC METHODS

Chirag Wadhwa (14)
Suraj Dev Deka (51)

Overview

Content:

- The Steiner Tree Problem (STP) is a classic NP-hard problem in graph theory & combinatorial optimization.
- Goal: Connect a given set of points (terminals) with minimum total cost.
- Allows adding extra nodes (Steiner points) to reduce cost.
- Applications:
    - Network routing
    - Circuit design
    - Phylogenetic trees
    - VLSI layout optimization

PROBLEM STATEMENT

Definition:

Given:

- A weighted undirected graph G = (V, E) with edge weights w(e)
- A set of terminal vertices T ⊆ V

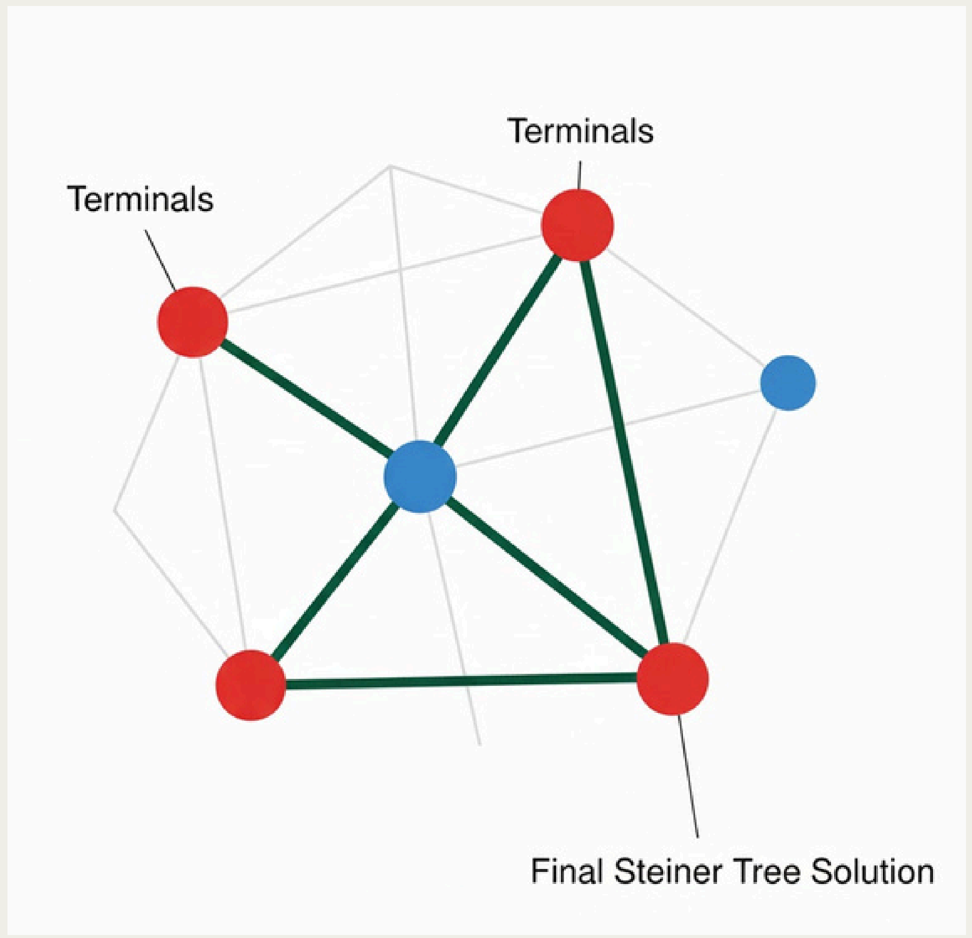
Find:

- A tree S = (V′, E′) that:
  1. Connects all terminals (T ⊆ V′)
  2. Minimizes total edge weight Σ w(e)

Formula:

$$\min \sum_{e \in E'} w(e) \quad \text{s.t. all terminals in T are connected}$$

Eg:



Red: The red nodes are the Terminals
Green: The dark green lines represent the final Steiner Tree, which is the solution network.
Blue: The blue nodes are the other optional Steiner points that were available in the graph but were not used in this particular solution.

NP-HARDNESS

- **Why NP-Hard:** No known polynomial-time algorithm can guarantee the optimal Steiner tree for all instances.
- Proven NP-complete (Decision version) by Karp (1972).
- Later strengthened by Garey, Graham & Johnson (1977).
- Both graph and geometric versions are NP-hard**.**

References:

1. Karp, R. M. (1972). Reducibility among combinatorial problems.
2. Garey, Graham, Johnson (1977). The complexity of computing Steiner minimal trees.

BRUTE FORCE ALGORITHM

Idea:

- Enumerate all subsets of potential Steiner nodes.
- For each subset, compute the MST of terminals + subset.
- Choose the minimum-cost one.

Explanation: This exhaustive algorithm tries all possible combinations of Steiner nodes, computes the Minimum Spanning Tree (MST) for each subset, and returns the lowest-cost one.
It guarantees the optimal Steiner tree but suffers from exponential time complexity **O(2^n)**, making it feasible only for small graphs (≤ 20 vertices).
The runtime grows dramatically — confirming the NP-hard nature of the problem.

Pseudocode:

```
Algorithm BruteForceSteinerTree(G, T)
  bestCost ← ∞
  bestTree ← ∅
  S ← V \ T
  for each subset X ⊆ S do
      V' ← T ∪ X
      G' ← induced subgraph of G on V'
      if G' connected then
          MST ← MinimumSpanningTree(G')
          cost ← weight(MST)
          if cost < bestCost then
              bestCost ← cost
              bestTree ← MST
          end if
      end if
  end for
  return bestTree, bestCost
```

APPROXIMATION ALGORITHM (TAKAHASHI & MATSUYAMA, 1980)

High-level idea:

- Start with one terminal.
- Iteratively connect the nearest unconnected terminal to the growing tree (like Prim's algorithm).
- Continue until all terminals are connected.

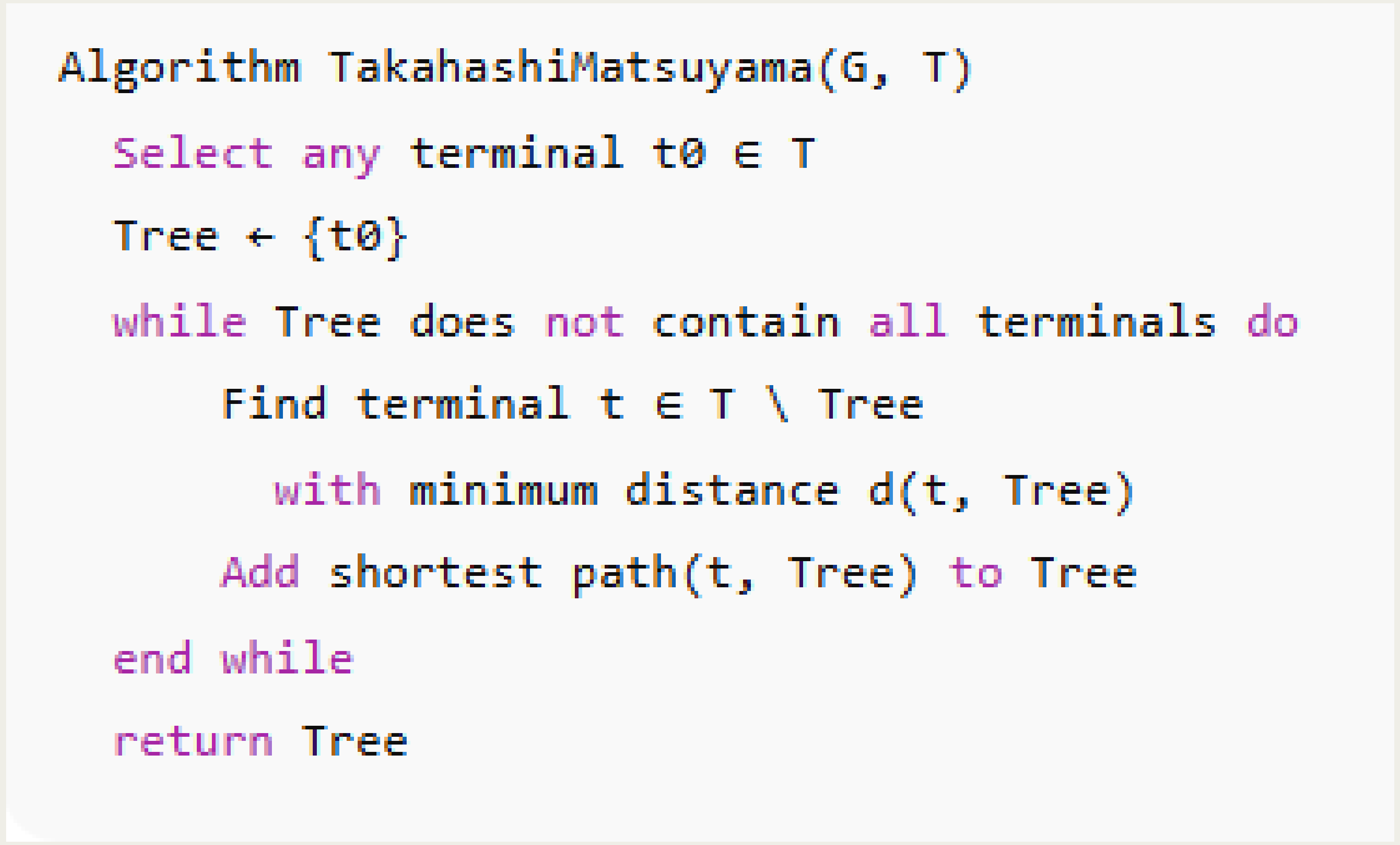
Guarantee:

- Cost ≤ 2 × OPT
- Practical for medium to large graphs
- Runs in polynomial time

Formula: $\mathrm{Cost}_{TM} \leq 2 \cdot (1 - \frac{1}{k}) \cdot OPT$

Explanation: This greedy approximation algorithm iteratively grows a tree starting from one terminal, always connecting the nearest unconnected terminal to the current tree using the shortest path in the original graph.
It runs in **O(|V|²)** and guarantees a solution within **2× the optimal cost.**

PseudoCode:



```
Algorithm TakahashiMatsuyama(G, T)
  Select any terminal t0 ∈ T
  Tree ← {t0}
  while Tree does not contain all terminals do
      Find terminal t ∈ T \ Tree
        with minimum distance d(t, Tree)
      Add shortest path(t, Tree) to Tree
  end while
  return Tree
```

HEURISTIC ALGORITHM (MST-PRUNING)

Idea:

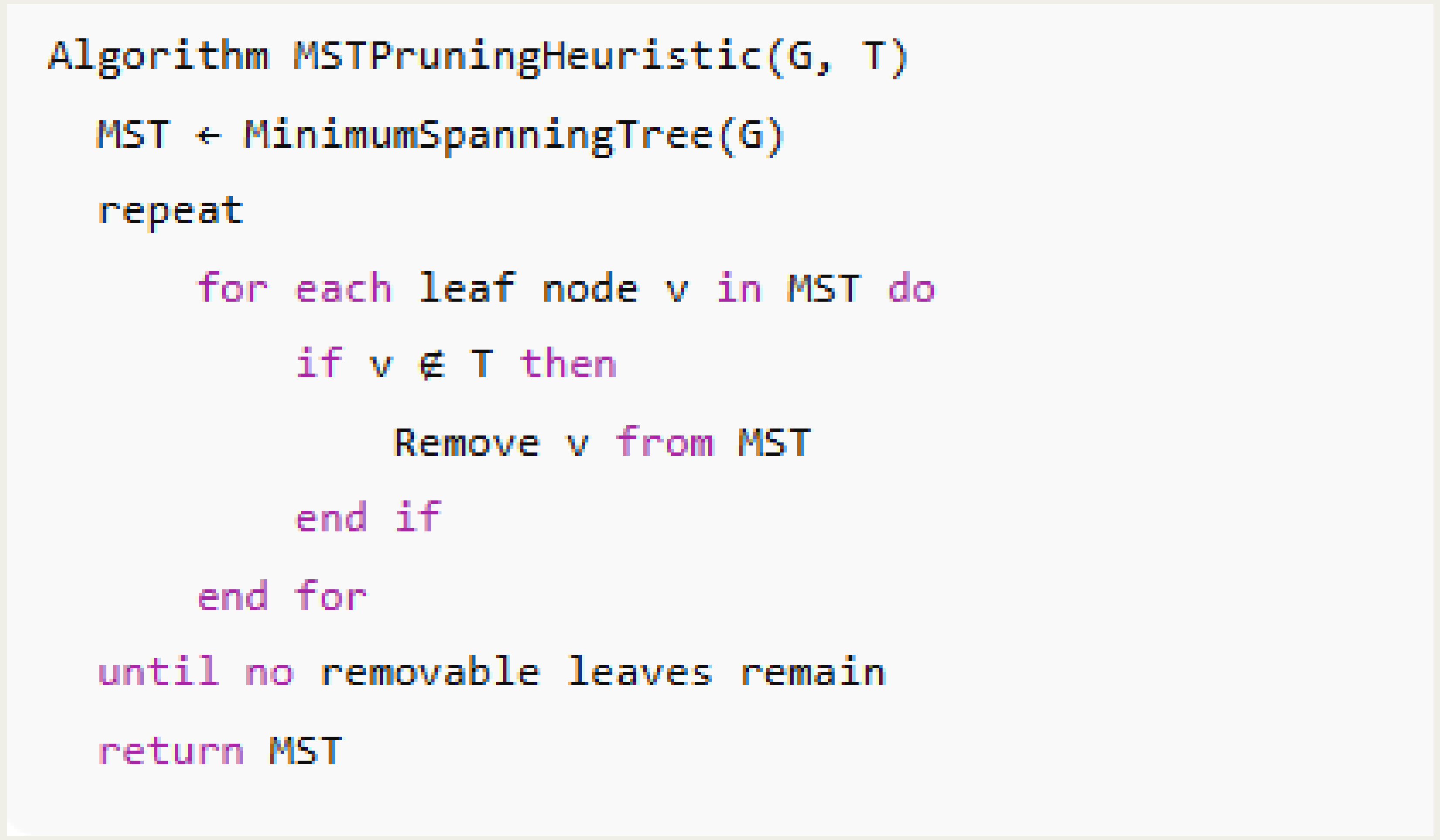
1. Compute MST of the entire graph.
2. Remove non-terminal leaf nodes iteratively.
3. Resulting tree connects all terminals cheaply.

When to use:

- For large datasets where approximation or brute force is slow.

Explanation: This heuristic builds an MST (Minimum Spanning Tree) over all vertices and then removes unnecessary non-terminal leaves.
It's extremely fast and simple — though not guaranteed to be optimal.
Complexity ≈ **O(|E| log |V|).**

PseudoCode:



```
Algorithm MSTPruningHeuristic(G, T)
  MST ← MinimumSpanningTree(G)
  repeat
      for each leaf node v in MST do
          if v ∉ T then
              Remove v from MST
          end if
      end for
  until no removable leaves remain
  return MST
```

EXPERIMENTAL SETUP

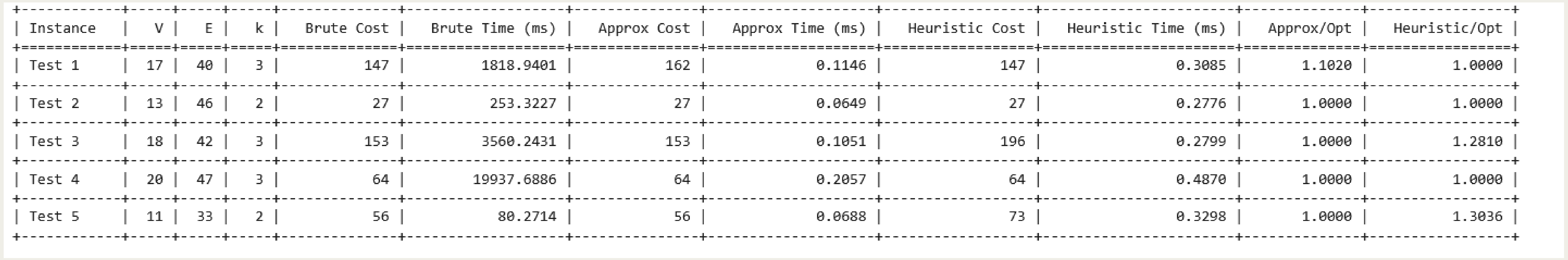
- Synthetic graphs generated for three scales — Small, Medium, and Large.
- Each instance includes random edge weights and terminals (≈ 10–20 % of total vertices).

- Metrics evaluated:
  - **Cost (solution quality)**
  - **Runtime (efficiency)**
  - **Approximation ratio (App/Opt, Heu/Opt)**

- Data Generation: All algorithms were tested on the same pre-generated graphs, which were saved as JSON files. The graphs were created as follows:
  - Connectivity: A random spanning tree was built first to guarantee the graph was connected.
  - Density: Additional random edges were added until the target edge count was reached.
  - Terminals: A percentage of vertices were randomly selected as terminals (minimum of 2).
  - Weights: All edge weights were assigned a random integer between 1 and 100.

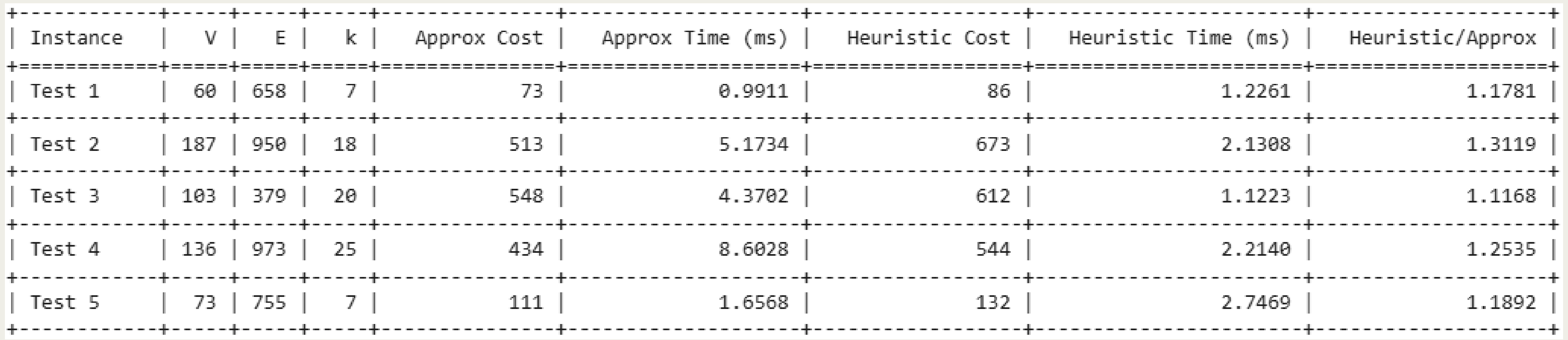
RESULTS

Small Graphs:



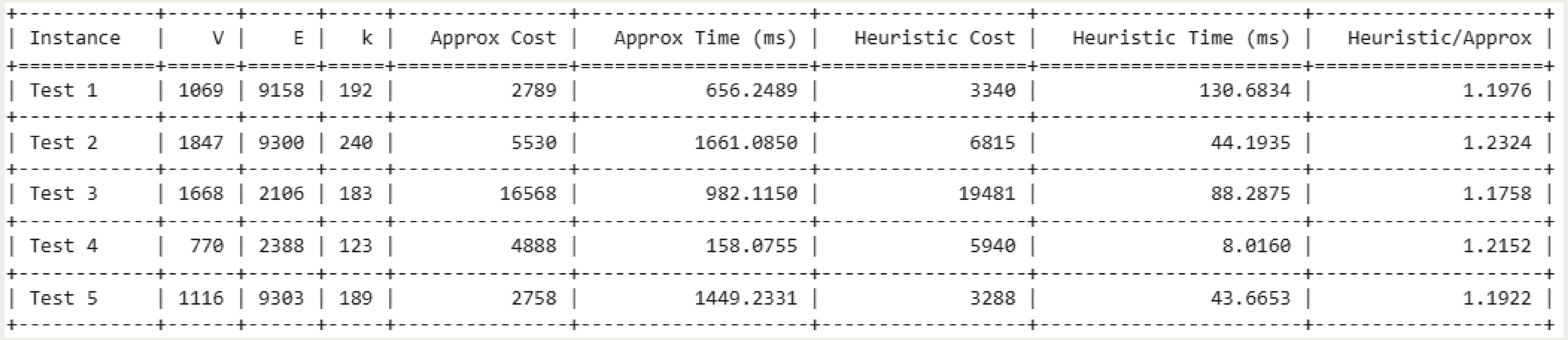
```
+-------------+-----+-----+-----+--------------+-------------------+--------------+-------------------+-----------------+----------------------+--------------+-----------------+
| Instance    |   V |   E |   k |   Brute Cost |   Brute Time (ms) |   Approx Cost |   Approx Time (ms) |   Heuristic Cost |   Heuristic Time (ms) |   Approx/Opt |   Heuristic/Opt |
+=============+=====+=====+=====+==============+===================+==============+===================+=================+======================+==============+=================+
| Test 1      |  17 |  40 |   3 |          147 |         1818.9401 |          162 |            0.1146 |             147 |               0.3085 |       1.1020 |          1.0000 |
+-------------+-----+-----+-----+--------------+-------------------+--------------+-------------------+-----------------+----------------------+--------------+-----------------+
| Test 2      |  13 |  46 |   2 |           27 |          253.3227 |           27 |            0.0649 |              27 |               0.2776 |       1.0000 |          1.0000 |
+-------------+-----+-----+-----+--------------+-------------------+--------------+-------------------+-----------------+----------------------+--------------+-----------------+
| Test 3      |  18 |  42 |   3 |          153 |         3560.2431 |          153 |            0.1051 |             196 |               0.2799 |       1.0000 |          1.2810 |
+-------------+-----+-----+-----+--------------+-------------------+--------------+-------------------+-----------------+----------------------+--------------+-----------------+
| Test 4      |  20 |  47 |   3 |           64 |        19937.6886 |           64 |            0.2057 |              64 |               0.4870 |       1.0000 |          1.0000 |
+-------------+-----+-----+-----+--------------+-------------------+--------------+-------------------+-----------------+----------------------+--------------+-----------------+
| Test 5      |  11 |  33 |   2 |           56 |           80.2714 |           56 |            0.0688 |              73 |               0.3298 |       1.0000 |          1.3036 |
+-------------+-----+-----+-----+--------------+-------------------+--------------+-------------------+-----------------+----------------------+--------------+-----------------+
```

Medium Graphs:



```
+-------------+-----+-----+-----+---------------+--------------------+------------------+----------------------+--------------------+
| Instance    |   V |   E |   k |   Approx Cost |   Approx Time (ms) |   Heuristic Cost |   Heuristic Time (ms) |   Heuristic/Approx |
+=============+=====+=====+=====+===============+====================+==================+======================+====================+
| Test 1      |  60 | 658 |   7 |            73 |             0.9911 |               86 |               1.2261 |             1.1781 |
+-------------+-----+-----+-----+---------------+--------------------+------------------+----------------------+--------------------+
| Test 2      | 187 | 950 |  18 |           513 |             5.1734 |              673 |               2.1308 |             1.3119 |
+-------------+-----+-----+-----+---------------+--------------------+------------------+----------------------+--------------------+
| Test 3      | 103 | 379 |  20 |           548 |             4.3702 |              612 |               1.1223 |             1.1168 |
+-------------+-----+-----+-----+---------------+--------------------+------------------+----------------------+--------------------+
| Test 4      | 136 | 973 |  25 |           434 |             8.6028 |              544 |               2.2140 |             1.2535 |
+-------------+-----+-----+-----+---------------+--------------------+------------------+----------------------+--------------------+
| Test 5      |  73 | 755 |   7 |           111 |             1.6568 |              132 |               2.7469 |             1.1892 |
+-------------+-----+-----+-----+---------------+--------------------+------------------+----------------------+--------------------+
```

Large Graphs:



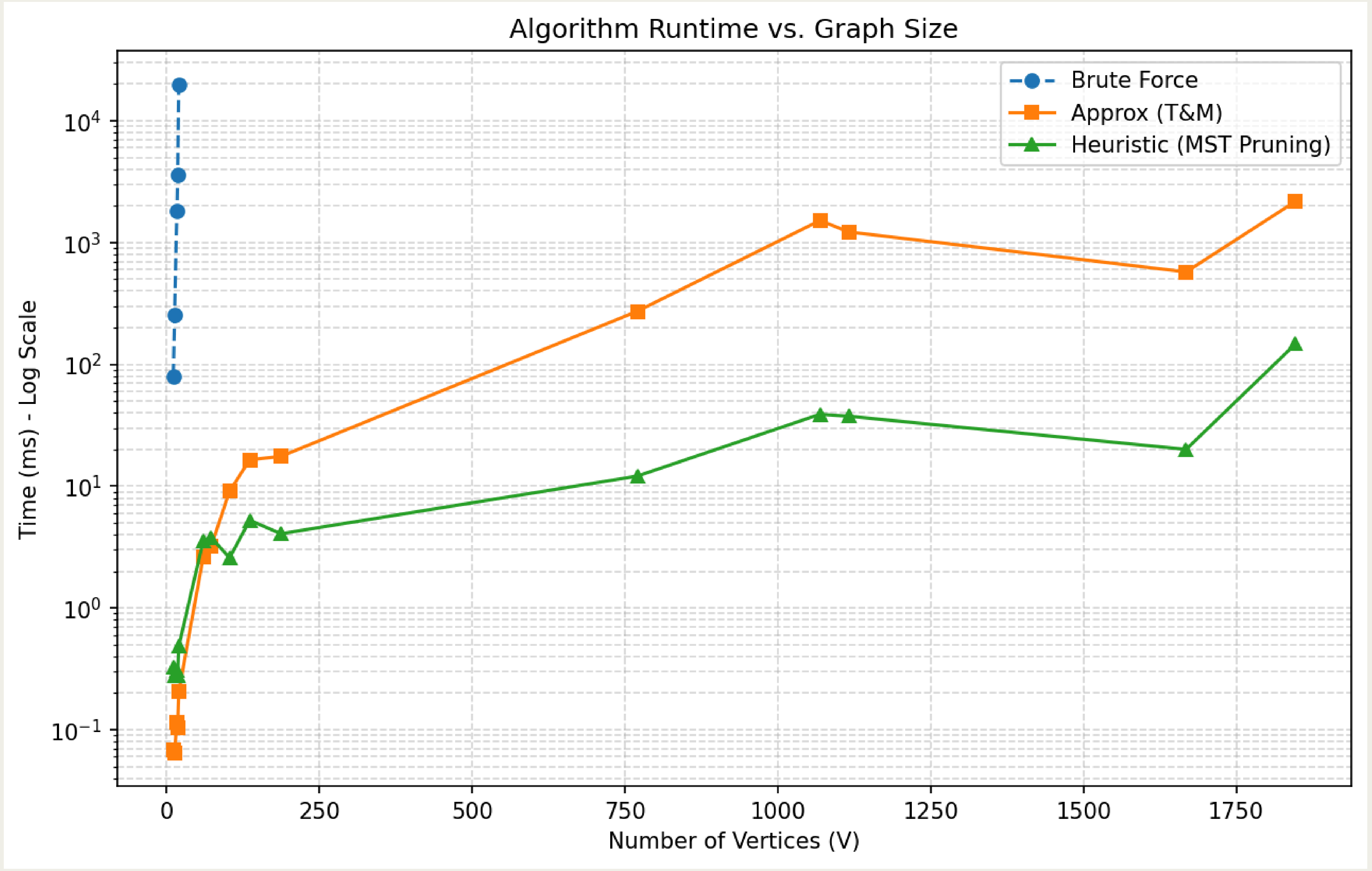
```
+-------------+------+------+-----+---------------+--------------------+------------------+----------------------+--------------------+
| Instance    |    V |    E |   k |   Approx Cost |   Approx Time (ms) |   Heuristic Cost |   Heuristic Time (ms) |   Heuristic/Approx |
+=============+======+======+=====+===============+====================+==================+======================+====================+
| Test 1      | 1069 | 9158 | 192 |          2789 |           656.2489 |             3340 |             130.6834 |             1.1976 |
+-------------+------+------+-----+---------------+--------------------+------------------+----------------------+--------------------+
| Test 2      | 1847 | 9300 | 240 |          5530 |          1661.0850 |             6815 |              44.1935 |             1.2324 |
+-------------+------+------+-----+---------------+--------------------+------------------+----------------------+--------------------+
| Test 3      | 1668 | 2106 | 183 |         16568 |           982.1150 |            19481 |              88.2875 |             1.1758 |
+-------------+------+------+-----+---------------+--------------------+------------------+----------------------+--------------------+
| Test 4      |  770 | 2388 | 123 |          4888 |           158.0755 |             5940 |               8.0160 |             1.2152 |
+-------------+------+------+-----+---------------+--------------------+------------------+----------------------+--------------------+
| Test 5      | 1116 | 9303 | 189 |          2758 |          1449.2331 |             3288 |              43.6653 |             1.1922 |
+-------------+------+------+-----+---------------+--------------------+------------------+----------------------+--------------------+
```

RUNTIME COMPARISON: BRUTE FORCE VS APPROXIMATION (T&M) VS HEURISTIC

Purpose: Evaluate scalability across small → medium → large graphs.

Key Points:

- **Brute Force:** grows exponentially; infeasible beyond 20 nodes (≈ 20 s runtime).
- **Approx (T&M)**: polynomial growth; handles 1000+ nodes within ~1–2 s.
- **Heuristic:** consistently fastest, even for > 1000 nodes (< 50 ms).

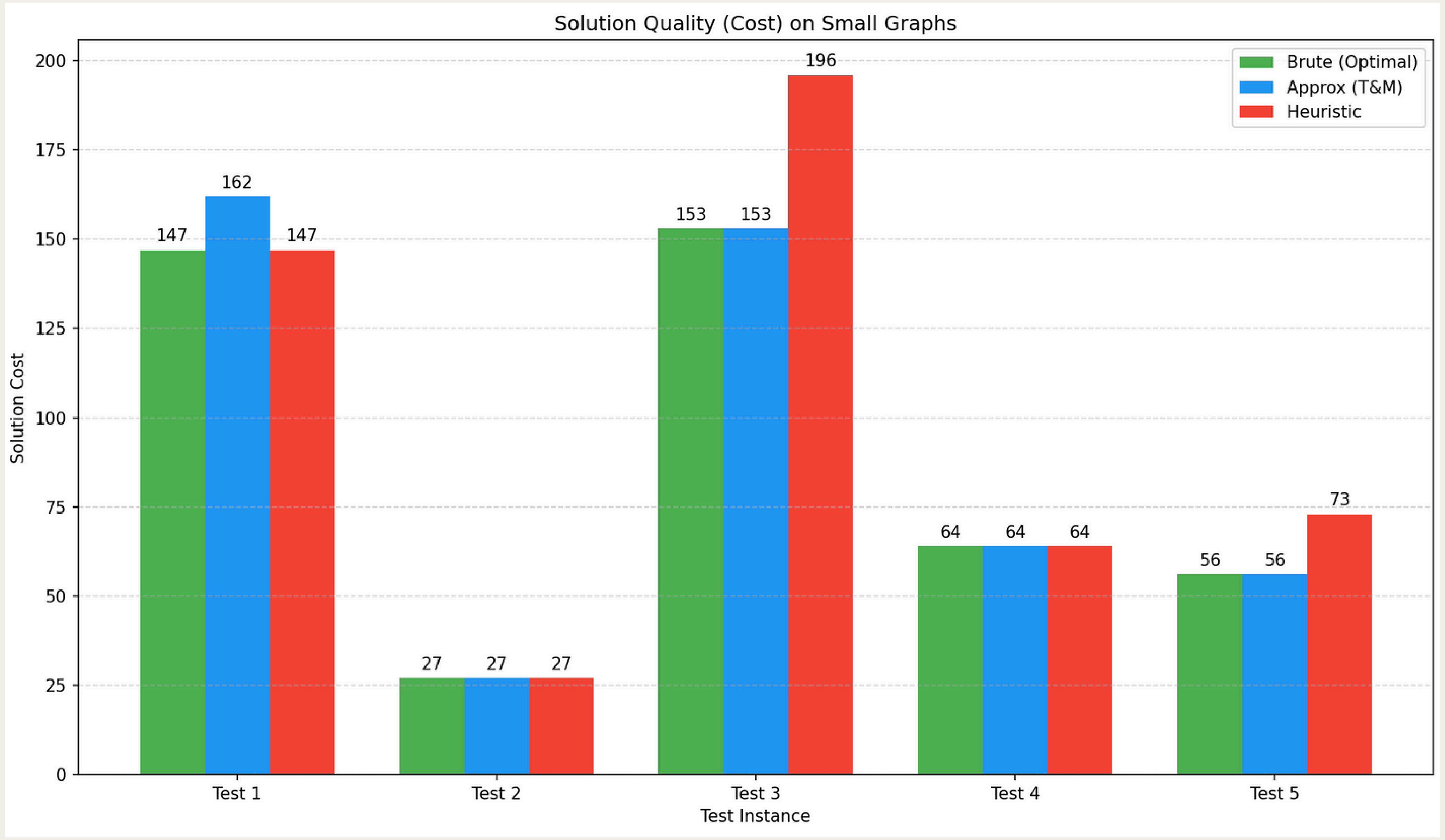


COST COMPARISON: APPROX VS HEURISTIC VS OPTIMAL

Purpose: Show how close each algorithm's cost is to the optimal (Brute Force baseline).

Key Points:

- **For small graphs,** both Approx and Heuristic match optimal closely.
- **For medium/large**, T&M gives lower cost trees (often within 5–15% of optimal).
- **Heuristic trees** are cheaper to compute but can be 15–30% more expensive.



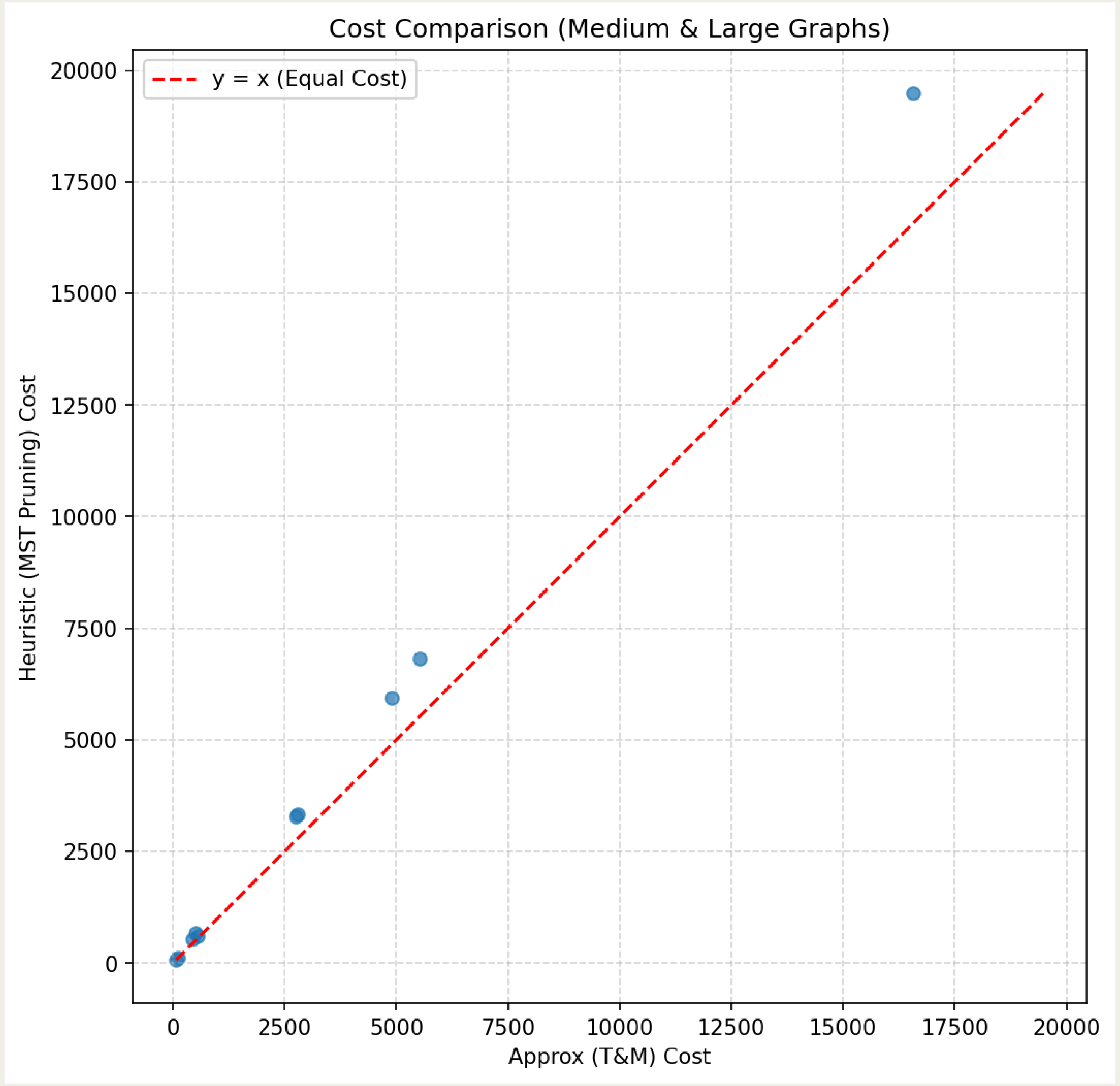
TRADE-OFF ANALYSIS AND OVERALL COMPARISON

Purpose: Summarize the global performance pattern.
Key Talking Points:

- **Brute Force**: Benchmark only; infeasible beyond small graphs.
- **Approx (T&M)**: Best balance of accuracy and speed.
- **Heuristic (MST-Pruning)**: Fastest, slightly less accurate.

The analysis uses two separate plots for a clear comparison:

- **For Small Graphs:** The algorithms are compared against the known perfect solution (found by the brute-force method). This measures their absolute quality.
- **For Medium & Large Graphs**: Since the perfect solution can't be found (brute force is too slow), the algorithms are compared against each other. This measures their relative quality on more realistic problems.



RESULT SUMMARY

- **Brute Force:**
  - Delivers the optimal Steiner tree but is exponentially slow.
  - Runtime jumps from 80 ms (11 vertices) to ~20 s (20 vertices) → infeasible for larger instances.
- **Takahashi–Matsuyama (Approximation):**
  - Found near-optimal or perfectly optimal solutions in most small cases.
  - Scales efficiently — solves ~1800-vertex graphs in 1–2 seconds.
  - Consistently gives lower-cost trees than the heuristic.
- **MST-Pruning (Heuristic):**
  - Fastest method overall — even largest graphs solved in <150 ms.
  - Produces slightly higher-cost trees (up to 30% above optimal).
  - Best suited for very large datasets where speed is critical.
- **Overall Trend:**
  - Small graphs: Approximation matches or beats heuristic quality.
  - Medium & large graphs: Clear speed–accuracy trade-off →
  - Heuristic = Fastest, Approximation = Most Accurate.
  - Brute Force only works as a baseline for small instances.

CONCLUSION

- The Steiner Tree Problem is NP-hard.
- Brute Force is infeasible for real-world graphs.
- Takahashi–Matsuyama offers a 2-approximation guarantee and good scalability.
- MST-pruning provides a fast heuristic for large graphs.
- Balancing accuracy vs efficiency is key.

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