Multi-level Steiner Trees

REYAN AHMED, University of Arizona, USA
PATRIZIO ANGELINI, Universität Tübingen, Germany
FARYAD DARABI SAHNEH, ALON EFRAT, and DAVID GLICKENSTEIN,
University of Arizona, USA
MARTIN GRONEMANN, Universität zu Köln, Germany
NIKLAS HEINSOHN, Universität Tübingen, Germany
STEPHEN G. KOBOUROV, RICHARD SPENCE, and JOSEPH WATKINS,
University of Arizona, USA
ALEXANDER WOLFF, Universität Würzburg, Germany

In the classical Steiner tree problem, given an undirected, connected graph $G = (V, E)$ with non-negative edge costs and a set of *terminals* $T \subseteq V$, the objective is to find a minimum-cost tree $E' \subseteq E$ that spans the terminals. The problem is APX-hard; the best-known approximation algorithm has a ratio of $\rho = \ln(4) + \varepsilon < 1.39$. In this article, we study a natural generalization, the *multi-level Steiner tree* (MLST) problem: Given a nested sequence of terminals $T_\ell \subset \cdots \subset T_1 \subseteq V$, compute nested trees $E_\ell \subseteq \cdots \subseteq E_1 \subseteq E$ that span the corresponding terminal sets with minimum total cost.

The MLST problem and variants thereof have been studied under various names, including Multi-level Network Design, Quality-of-Service Multicast tree, Grade-of-Service Steiner tree, and Multi-tier tree. Several approximation results are known. We first present two simple $O(\ell)$-approximation heuristics. Based on these, we introduce a rudimentary composite algorithm that generalizes the above heuristics, and determine its approximation ratio by solving a linear program. We then present a method that guarantees the same approximation ratio using at most 2ℓ Steiner tree computations. We compare these heuristics experimentally on various instances of up to 500 vertices using three different network generation models. We also present several integer linear programming formulations for the MLST problem and compare their running times on these instances. To our knowledge, the composite algorithm achieves the best approximation ratio for up to $\ell = 100$ levels, which is sufficient for most applications, such as network visualization or designing multi-level infrastructure.

This work is partially supported by NSF Grants No. CCF-1423411 and No. CCF-1712119
Authors' addresses: R. Ahmed, F. D. Sahneh, and R. Spence, Room 721, Department of Computer Science, Gould-Simpson Building, The University of Arizona, Tucson, AZ 85721-0077; emails: abureyanahmed@email.arizona.edu, faryad@cs.arizona.edu, rcspence@email.arizona.edu; P. Angelini and N. Heinsohn, Arbeitsbereich Algorithmik, Wilhelm-Schickard Institut für Informatik, Sand 14, D-72076 Tübingen, Deutschland; email: heinsohn@informatik.uni-tuebingen.de; A. Efrat, Room 742, Department of Computer Science, Gould-Simpson Building, The University of Arizona, Tucson, AZ 85721-0077; email: alon@cs.arizona.edu; D. Glickenstein, Room 204, Department of Mathematics, The University of Arizona, Tucson, AZ 85721-0089; email: glickenstein@math.arizona.edu; M. Gronemann, Universität zu Köln, Institut für Informatik, Weyertal 121, D-50931 Köln; email: gronemann@informatik.uni-koeln.de; S. G. Kobourov, Room 715, Department of Computer Science, Gould-Simpson Building, The University of Arizona, Tucson, AZ 85721-0077; email: kobourov@cs.arizona.edu; J. Watkins, Room S321, ENR2, The University of Arizona, Tucson, AZ 85719; email: jwatkins@math.arizona.edu; A. Wolff, Room E29, Lehrstuhl für Informatik I, Universität Würzburg Am Hubland, D-97074 Würzburg.

1084-6654/2019/12-ART2.5 $15.00
https://doi.org/10.1145/3368621

CCS Concepts: • **Theory of computation → Sparsification and spanners**;

Additional Key Words and Phrases: Approximation algorithm, Steiner tree, multi-level graph representation

ACM Reference format:
Reyan Ahmed, Patrizio Angelini, Faryad Darabi Sahneh, Alon Efrat, David Glickenstein, Martin Gronemann, Niklas Heinsohn, Stephen G. Kobourov, Richard Spence, Joseph Watkins, and Alexander Wolff. 2019. Multi-level Steiner Trees. *J. Exp. Algorithmics* 24, 2, Article 2.5 (December 2019), 22 pages.
https://doi.org/10.1145/3368621

1 INTRODUCTION

Let $G = (V, E)$ be an undirected, connected graph with positive edge costs $c : E \to \mathbb{R}^+$, and let $T \subseteq V$ be a set of vertices called *terminals*. A *Steiner tree* is a tree in G that spans T. The *network (graph) Steiner tree problem* (ST) is to find a minimum-cost Steiner tree $E' \subseteq E$, where the cost of E' is $c(E') = \sum_{e \in E'} c(e)$. ST is one of Karp's initial NP-hard problems [13]; see also a survey [22], an online compendium [12], and a textbook [19].

Due to its practical importance in many domains, there is a long history of exact and approximation algorithms for the problem. The classical 2-approximation algorithm for ST [11] uses the *metric closure* of G, i.e., the complete edge-weighted graph G^* with vertex set T in which, for every edge uv, the cost of uv equals the length of a shortest u–v path in G. A minimum spanning tree of G^* corresponds to a 2-approximate Steiner tree in G.

Currently, the last in a long list of improvements is the LP-based approximation algorithm of Byrka et al. [6], which has a ratio of $\ln(4) + \varepsilon < 1.39$. Their algorithm uses a new iterative randomized rounding technique. Note that ST is APX-hard [4]; more concretely, it is NP-hard to approximate the problem within a factor of 96/95 [8]. This is in contrast to the geometric variant of the problem, where terminals correspond to points in the Euclidean or rectilinear plane. Both variants admit polynomial-time approximation schemes (PTAS) [1, 16], while this is not true for the general metric case [4].

In this article, we consider the natural generalization of ST where the terminals appear on "levels" (or "grades of service") and must be connected by edges of appropriate levels. We propose new approximation algorithms and compare them to existing ones both theoretically and experimentally.

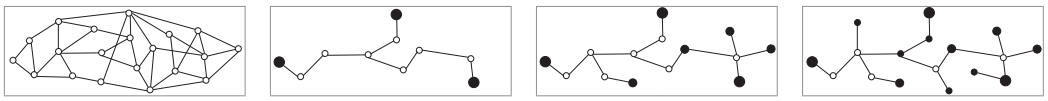
Definition 1.1 (Multi-level Steiner Tree (MLST) Problem). Given a connected, undirected graph $G = (V, E)$ with edge weights $c : E \to \mathbb{R}^+$ and ℓ nested terminal sets $T_\ell \subset \cdots \subset T_1 \subseteq V$, a *multi-level Steiner tree* consists of ℓ nested edge sets $E_\ell \subseteq \cdots \subseteq E_1 \subseteq E$ such that E_i spans T_i for all $1 \leq i \leq \ell$. The cost of an MLST is defined by the sum of the edge weights across all levels, $\sum_{i=1}^{\ell} c(E_i) = \sum_{i=1}^{\ell} \sum_{e \in E_i} c(e)$. The MLST problem is to find an MLST $E_{\mathrm{OPT},\ell} \subseteq \cdots \subseteq E_{\mathrm{OPT},1} \subseteq E$ with minimum cost.

Since the edge sets are nested, the cost of an MLST equivalently equals $\sum_{e \in E} L(e)c(e)$, where $L(e)$ denotes the highest level that edge e appears in, where $L(e) = 0$ if $e \notin E_1$. This emphasizes that the cost of each edge is multiplied by the number of levels it appears on.

We denote the cost of an optimal MLST by OPT. We can write

$$\mathrm{OPT} = \ell \mathrm{OPT}_\ell + (\ell - 1)\mathrm{OPT}_{\ell-1} + \cdots + \mathrm{OPT}_1,$$

where $\mathrm{OPT}_\ell = c(E_{\mathrm{OPT},\ell})$ and $\mathrm{OPT}_i = c(E_{\mathrm{OPT},i} \setminus E_{\mathrm{OPT},i+1})$ for $\ell - 1 \geq i \geq 1$. Thus, OPT_i represents the cost of edges on level i but not on level $i + 1$ in the minimum cost MLST. Figure 1 shows an example of an MLST for $\ell = 3$.



(a) (b) (c) (d)

Fig. 1. An MLST with $\ell = 3$ for the input graph (a). Filled and empty circles represent terminal and non-terminal nodes, respectively. Note that the level-3 tree (b) is contained in the level-2 tree (c), which is in turn contained in the level-1 tree (d).

Applications. This problem has natural applications in designing multi-level infrastructure of low cost. Apart from this application in network design, multi-scale representations of graphs are useful in applications such as network visualization, where the goal is to represent a given graph at different levels of detail.

Previous Work. Variants of the MLST problem have been studied previously under various names, such as *Multi-level Network Design (MLND)* [2], *Multi-tier Tree (MTT)* [15], *Quality-of-Service (*QoS*) Multicast Tree* [7], and *Priority-Steiner Tree* [9].

In MLND, the vertices of the given graph are partitioned into ℓ levels, and the task is to construct an ℓ-level network. Each edge $(i, j) \in E$ can contain one of ℓ different facility types (levels), each with a different cost (denoted "secondary" and "primary" with costs $0 \le b_{ij} \le a_{ij}$ for two levels). The vertices on each level must be connected by edges of the corresponding level or higher, and edges of higher level are more costly. The cost of an edge partition is the sum of all edge costs, and the task is to find a partition of minimum cost. Let ρ be the ratio of the best approximation algorithm for (single-level) ST, that is, currently $\rho = \ln(4) + \varepsilon < 1.39$. Balakrishnan et al. [2] gave a $(4/3)\rho$-approximation algorithm for two-level MLND with proportional edge costs. Note that the definitions of MLND and MLST treat the bottom level differently. While MLND requires that *all* vertices are connected eventually, this is not the case for MLST.

For MTT, which is equivalent to MLND, Mirchandani [15] presented a recursive algorithm that involves 2^ℓ Steiner tree computations. For $\ell = 3$, the algorithm achieves an approximation ratio of 1.522ρ independently of the edge costs $c^1, \ldots, c^\ell : E \to \mathbb{R}^+$. For proportional edge costs, Mirchandani's analysis yields even an approximation ratio of 1.5ρ for $\ell = 3$. Recall, however, that this assumes $T_1 = V$, and setting the edge costs on the bottom level to zero means that edge costs are *not* proportional.

In the QoS Multicast Tree problem [7] one is given a graph, a source vertex s, and a level between 1 and k for each terminal (1 for highest priority). The task is to find a minimum-cost Steiner tree that connects all terminals to s. The level of an edge e in this tree is the minimum over the levels of the terminals that are connected to s via e. The cost of the edges and of the tree are as above. As a special case, Charikar et al. [7] studied the *rate model*, where edge costs are proportional, and show that the problem remains NP-hard if all vertices (except the source) are terminals (at some level). Note that if we choose as source any vertex at the top level T_ℓ, then MLST can be seen as an instance of the rate model.

Charikar et al. [7] gave a simple 4ρ-approximation algorithm for the rate model. Given an instance φ, their algorithm constructs an instance φ' where the levels of all vertices are rounded up to the nearest power of 2. Then the algorithm simply computes a Steiner tree at each level of φ' and prunes the union of these Steiner trees into a single tree. The ratio can be improved to $e\rho$, where e is the base of the natural logarithm, using randomized doubling.

Instead of taking the union of the Steiner trees on each rounded level, Karpinski et al. [14] contract them into the source in each step, which yields a 2.454ρ-approximation. They also gave a $(1.265 + \varepsilon)\rho$-approximation for the two-level case. (Since these results are not stated with respect to ρ, but depend on several Steiner tree approximation algorithms—among them the best

approximation algorithm with ratio 1.549 [20] available at the time—we obtained the numbers given here by dividing their results by 1.549 and stating the factor ρ.)

For the more general Priority-Steiner Tree problem, where edge costs are not necessarily proportional, Charikar et al. [7] gave a $\min\{2 \ln |T|, \ell\rho\}$-approximation algorithm. Chuzhoy et al. [9] showed that Priority-Steiner Tree does not admit an $O(\log \log n)$-approximation algorithm unless $\text{NP} \subseteq \text{DTIME}(n^{O(\log \log \log n)})$. For Euclidean MLST, Xue et al. [23] gave a recursive algorithm that uses any algorithm for Euclidean Steiner Tree (EST) as a subroutine. With a PTAS [1, 16] for EST, the approximation ratio of their algorithm is $4/3 + \varepsilon$ for $\ell = 2$ and $(5 + 4\sqrt{2})/7 + \varepsilon \approx 1.522 + \varepsilon$ for $\ell = 3$.

Our Contribution. We give two simple approximation algorithms for MLST, bottom-up and top-down, in Section 2.1. The bottom-up heuristic uses a Steiner tree at the bottom level for the higher levels after pruning unnecessary edges at each level. The top-down heuristic first computes a Steiner tree on the top level. Then it passes edges down from level to level until the bottom level terminals are spanned.

In Section 2.2, we propose a composite heuristic that generalizes these, by examining all possible $2^{\ell-1}$ (partial) top-down and bottom-up combinations and returning the one with the lowest cost. We propose a linear program that finds the approximation ratio of the composite heuristic for any fixed value of ℓ, and compute approximation ratios for up to $\ell = 100$ levels, which turn out to be better than those of previously known algorithms. However, the composite heuristic requires roughly $2^\ell \ell$ ST computations.

Therefore, we propose a procedure that achieves the same approximation ratio as the composite heuristic but needs at most 2ℓ ST computations. In particular, it achieves a ratio of 1.5ρ for $\ell = 3$ levels, which settles a question posed by Karpinski et al. [14] who were asking whether the $(1.522 + \varepsilon)$-approximation of Xue et al. [23] can be improved for $\ell = 3$. Note that Xue et al. treated the Euclidean case, so their ratio does not include the factor ρ. We generalize an integer linear programming (ILP) formulation for ST [18] to obtain an ILP formulation for the MLST problem in Section 3. We experimentally evaluate several approximation and exact algorithms on a wide range of problem instances in Section 4, and discuss an application of this problem in visualizing large networks. The results show that the new algorithms are also surprisingly good in practice. We conclude in Section 5.

2 APPROXIMATION ALGORITHMS

In this section, we propose several approximation algorithms for the MLST problem.

In Section 2.1, we show that the natural approach of computing edge sets either from top to bottom or vice versa, already yields $O(\ell)$-approximations; we call these two approaches *top-down* and *bottom-up*, and denote their costs by TOP and BOT, respectively. Then, we show that running the two approaches and selecting the solution with minimum cost produces a better approximation ratio than either top-down or bottom-up.

In Section 2.2, we propose a composite approach that mixes the top-down and bottom-up approaches by solving ST on a certain subset of levels, then propagating the chosen edges to higher and lower levels in a way similar to the previous approaches. We then run the algorithm for each of the $2^{\ell-1}$ possible subsets, and select the solution with minimum cost. For all practically relevant values of ℓ ($\ell \leq 100$), our results improve over the state of the art.

2.1 Top-down and Bottom-up Approaches

We present top-down and bottom-up approaches for computing approximate multi-level Steiner trees. The approaches are similar to the MST and Forward Steiner Tree (FST) heuristics by Balakrishnan et al. [2]; however, we generalize the analysis to an arbitrary number of levels.

In the top-down approach, an exact or approximate Steiner tree $E_{\mathrm{TOP},\ell}$ spanning the top level T_ℓ is computed. Then, we modify the edge weights by setting $c(e) := 0$ for every edge $e \in E_{\mathrm{TOP},\ell}$. In the resulting graph, we compute a Steiner tree $E_{\mathrm{TOP},\ell-1}$ spanning the terminals in $T_{\ell-1}$. This extends $E_{\mathrm{TOP},\ell}$ in a greedy way to span the terminals in $T_{\ell-1}$ not already spanned by $E_{\mathrm{TOP},\ell}$. Iterating this procedure for all levels yields a solution $E_{\mathrm{TOP},\ell} \subseteq \cdots \subseteq E_{\mathrm{TOP},1} \subseteq E$ with cost TOP.

In the bottom-up approach, a Steiner tree $E_{\mathrm{BOT},1}$ spanning the bottom level T_1 is computed. This induces a valid solution for all levels. We can "prune" edges by letting $E_{\mathrm{BOT},i}$ be the smallest subtree of $E_{\mathrm{BOT},1}$ that spans all the terminals in T_i, giving a solution with cost BOT. Note that the top-down and bottom-up approaches involve ℓ and 1 Steiner tree computations, respectively.

A natural approach is to run both top-down and bottom-up approaches and select the solution with minimum cost. This yields an approximation ratio better than those from top-down or bottom-up. Let $\rho \geq 1$ denote the approximation ratio for ST (that is, $\rho = 1$ corresponds to using an exact ST subroutine). Let MIN_i denote the cost of a minimum Steiner tree over the terminal set T_i with original edge weights, independently of other levels, so that $\mathrm{MIN}_\ell \leq \mathrm{MIN}_{\ell-1} \leq \cdots \leq \mathrm{MIN}_1$. Then $\mathrm{OPT} \geq \sum_{i=1}^{\ell} \mathrm{MIN}_i$ trivially.

Theorem 2.1. *For $\ell \geq 2$ levels, the top-down approach is an $\frac{\ell+1}{2}\rho$-approximation to the MLST problem, the bottom-up approach is an $\ell\rho$-approximation, and the algorithm returning the minimum of* TOP *and* BOT *is an $\frac{\ell+2}{3}\rho$-approximation.*

In the following, we give the proof of Theorem 2.1. Let TOP be the total cost produced by the top-down approach, and let $\mathrm{TOP}_i = c(E_{\mathrm{TOP},i} \setminus E_{\mathrm{TOP},i+1})$ be the cost of edges on level i but not on level $i+1$. Then $\mathrm{TOP} = \sum_{i=1}^{\ell} i\mathrm{TOP}_i$. Define BOT and BOT_i analogously.

Lemma 2.2. *The following inequalities relate* TOP *with* OPT*:*

$$\mathrm{TOP}_\ell \leq \rho\mathrm{OPT}_\ell, \tag{2.1}$$

$$\mathrm{TOP}_{\ell-i} \leq \rho(\mathrm{OPT}_{\ell-i} + \cdots + \mathrm{OPT}_\ell) \text{ for all } 1 \leq i \leq \ell - 1. \tag{2.2}$$

Proof. Inequality Equation (2.1) follows from the fact that $E_{\mathrm{TOP},\ell}$ is a ρ-approximation for ST over T_ℓ, that is, $\mathrm{TOP}_\ell \leq \rho\mathrm{MIN}_\ell \leq \rho\mathrm{OPT}_\ell$. To show Equation (2.2), note that $\mathrm{TOP}_{\ell-i}$ represents the cost of the Steiner tree over terminals $T_{\ell-i}$ with some edges (those already included in $E_{\ell-i+1}$) having weight $c(e)$ set to zero. Then $\mathrm{TOP}_{\ell-i} \leq \rho\mathrm{MIN}_{\ell-i}$. Since $E_{\mathrm{OPT},\ell-i}$ spans $T_{\ell-i}$ by definition, we have $\mathrm{MIN}_{\ell-i} \leq c(E_{\mathrm{OPT},\ell-i}) = \mathrm{OPT}_{\ell-i} + \cdots + \mathrm{OPT}_\ell$. By transitivity, $\mathrm{TOP}_{\ell-i} \leq \rho(\mathrm{OPT}_{\ell-i} + \cdots + \mathrm{OPT}_\ell)$ as desired. □

Using Lemma 2.2 yields an upper bound on TOP in terms of $\mathrm{OPT}_1, \ldots, \mathrm{OPT}_\ell$:

$$\begin{aligned}
\mathrm{TOP} &= \ell\mathrm{TOP}_\ell + (\ell-1)\mathrm{TOP}_{\ell-1} + \cdots + \mathrm{TOP}_1 \\
&\leq \ell\rho\mathrm{OPT}_\ell + (\ell-1)\rho(\mathrm{OPT}_{\ell-1} + \mathrm{OPT}_\ell) + \cdots + \rho(\mathrm{OPT}_1 + \mathrm{OPT}_2 + \cdots + \mathrm{OPT}_\ell) \\
&= \rho\left(\frac{(\ell+1)\ell}{2}\mathrm{OPT}_\ell + \frac{\ell(\ell-1)}{2}\mathrm{OPT}_{\ell-1} + \cdots + \frac{2\cdot 1}{2}\mathrm{OPT}_1\right) \\
&\leq \frac{\ell+1}{2}\rho \cdot \mathrm{OPT}.
\end{aligned}$$

Therefore, the top-down approach is an $\frac{\ell+1}{2}\rho$-approximation. In Figure 2, we provide an example showing that our analysis is tight for $\rho = 1$.

The bottom-up approach is a fairly trivial $\ell\rho$-approximation to the MLST problem, even without pruning edges. Consequentially, $\mathrm{BOT} \leq \ell \cdot c(E_{\mathrm{BOT},1})$ as pruning no edges results in a solution with cost $\ell \cdot c(E_{\mathrm{BOT},1})$.

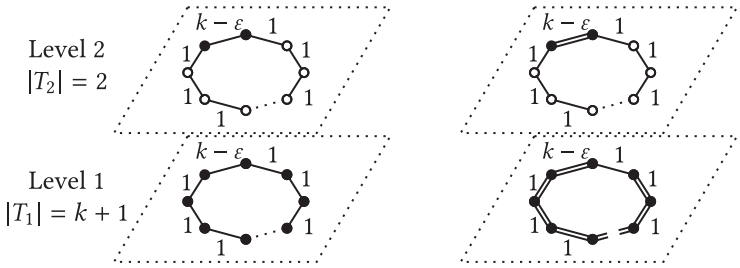


Fig. 2. The approximation ratio of $\frac{\ell+1}{2}$ for the top-down approach is asymptotically tight. In the example above for $\ell = 2$, the input graph (left) consists of a $(k + 1)$-cycle with one edge of weight $k - \varepsilon$. The solution returned by top-down (right) has cost TOP $= (k - \varepsilon) + (k - \varepsilon + k) \approx 3k$, whereas OPT $= 2k$.

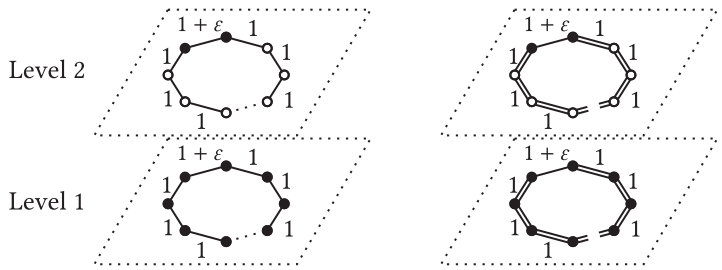


Fig. 3. The approximation ratio of ℓ for the bottom-up approach is asymptotically tight. Using the same input graph as in Figure 2, except by modifying the edge of weight $k - \varepsilon$ so that its weight is $1 + \varepsilon$, we see that BOT $= 2k$, whereas OPT $= (1 + \varepsilon) + (1 + \varepsilon + k - 1) \approx k + 1$.

As $E_{\mathrm{BOT},1}$ is found by computing a Steiner tree over the bottom level T_1, we have $c(E_{\mathrm{BOT},1}) \leq \rho \mathrm{MIN}_1$. Additionally, $\mathrm{MIN}_1 \leq c(E_{\mathrm{OPT},1}) = \mathrm{OPT}_1 + \mathrm{OPT}_2 + \cdots + \mathrm{OPT}_\ell$ as $E_{\mathrm{OPT},1}$ is necessarily a Steiner tree spanning T_1. Combining these inequalities yields

$$\begin{aligned}
\mathrm{BOT} &\leq \ell \cdot c(E_{\mathrm{BOT},1}) \\
&\leq \ell \rho \mathrm{MIN}_1 \\
&\leq \ell \rho (\mathrm{OPT}_1 + \mathrm{OPT}_2 + \cdots + \mathrm{OPT}_\ell) \\
&\leq \ell \rho \sum_{i=1}^{\ell} i \mathrm{OPT}_i \\
&= \ell \rho \cdot \mathrm{OPT}.
\end{aligned}$$

Again, the approximation ratio (for $\rho = 1$) is asymptotically tight; see Figure 3.

We show that taking the better of the two solutions returned by the top-down and the bottom-up approach provides a $\frac{4}{3}\rho$-approximation to MLST for $\ell = 2$. To prove this, we use the simple fact that $\min\{x, y\} \leq \alpha x + (1 - \alpha)y$ for all $x, y \in \mathbb{R}$ and $\alpha \in [0, 1]$. Using the previous results on the upper bounds for TOP and BOT for $\ell = 2$:

$$\begin{aligned}
\min\{\mathrm{TOP}, \mathrm{BOT}\} &\leq \alpha(3\rho\,\mathrm{OPT}_2 + \rho\,\mathrm{OPT}_1) + (1 - \alpha)(2\rho\,\mathrm{OPT}_2 + 2\rho\,\mathrm{OPT}_1) \\
&= (2 + \alpha)\rho\,\mathrm{OPT}_2 + (2 - \alpha)\rho\,\mathrm{OPT}_1.
\end{aligned}$$

Setting $\alpha = \frac{2}{3}$ gives $\min\{\mathrm{TOP}, \mathrm{BOT}\} \leq \frac{8}{3}\rho\,\mathrm{OPT}_2 + \frac{4}{3}\rho\,\mathrm{OPT}_1 = \frac{4}{3}\rho\,\mathrm{OPT}$.

For $\ell > 2$ levels, using the same idea gives

$$\begin{aligned}\min\{\text{TOP}, \text{BOT}\} &\leq \alpha\rho \sum_{i=1}^{\ell} \frac{i(i+1)}{2}\text{OPT}_i + (1-\alpha)\ell\rho \sum_{i=1}^{\ell} \text{OPT}_i \\ &= \sum_{i=1}^{\ell} \left[\left(\frac{i(i+1)}{2} - \ell\right)\alpha + \ell\right]\rho\text{OPT}_i.\end{aligned}$$

Since we are comparing $\min\{\text{TOP}, \text{BOT}\}$ to $t \cdot \text{OPT}$ for some approximation ratio $t > 1$, we can compare coefficients and find the smallest $t \geq 1$ such that the system of inequalities

$$\begin{aligned}\left(\frac{\ell(\ell+1)}{2} - \ell\right)\rho\alpha + \ell\rho &\leq \ell t \\ \left(\frac{(\ell-1)\ell}{2} - \ell\right)\rho\alpha + \ell\rho &\leq (\ell-1)t \\ &\vdots \\ \left(\frac{2\cdot 1}{2} - \ell\right)\rho\alpha + \ell\rho &\leq t\end{aligned}$$

has a solution $\alpha \in [0, 1]$. Adding the first inequality to $\ell/2$ times the last inequality yields $\frac{\ell^2+2\ell}{2}\rho \leq \frac{3\ell t}{2}$. This leads to $t \geq \frac{\ell+2}{3}\rho$. Also, it can be shown algebraically that $(t, \alpha) = (\frac{\ell+2}{3}\rho, \frac{2}{3})$ simultaneously satisfies the above inequalities. This implies that $\min\{\text{TOP}, \text{BOT}\} \leq \frac{\ell+2}{3}\rho \cdot \text{OPT}$ and concludes the proof of Theorem 2.1.

Combining the graphs in Figures 2 and 3 shows that our analysis of the combined top-down and bottom-up approaches (with ratio $\frac{4}{3}$) is asymptotically tight.

2.2 Composite Algorithm

We describe an approach that generalizes the above approaches to obtain a better approximation ratio for $\ell > 2$ levels. The main idea behind this composite approach is the following: In the top-down approach, we choose a set of edges $E_{\text{TOP},\ell}$ that spans T_ℓ, and then we propagate this choice to levels $\ell - 1, \ldots, 1$ by setting the cost of these edges to 0. However, in the bottom-up approach, we choose a set of edges $E_{\text{BOT},1}$ that spans T_1, which is propagated to levels $2, \ldots, \ell$ (possibly with some pruning of unneeded edges). The idea is that for $\ell > 2$, we can choose a set of intermediate levels and propagate our choices between these levels in a top-down manner, and to the levels lying in between them in a bottom-up manner.

Formally, let $Q = \{i_1, i_2, \ldots, i_m\}$ with $1 = i_1 < i_2 < \cdots < i_m \leq \ell$ be a subset of levels sorted in increasing order. We first compute a Steiner tree $E_{i_m} = ST(G, T_{i_m})$ on the highest level i_m, which induces trees $E_{i_m+1}, \ldots, E_\ell$ similar to the bottom-up approach. Then, we set the weights of edges in E_{i_m} in G to zero (as in the top-down approach) and compute a Steiner tree $E_{i_{m-1}} = ST(G, T_{i_{m-1}})$ for level i_{m-1} in the reweighed graph. Again, we can use $E_{i_{m-1}}$ to construct the trees $E_{i_{m-1}+1}, \ldots, E_{i_m-1}$. Repeating this procedure until spanning $E_{i_1} = E_1$ results in a valid solution to the MLST problem.

Note that the top-down and bottom-up heuristics are special cases of this approach, with $Q = \{1, 2, \ldots, \ell\}$ and $Q = \{1\}$, respectively. Figure 4 provides an illustration of how such a solution is computed in the top-down, bottom-up, and an arbitrary heuristic. Given $Q \subseteq \{1, 2, \ldots, \ell\}$, let $\text{CMP}(Q)$ be the cost of the MLST solution returned by the composite approach over Q.

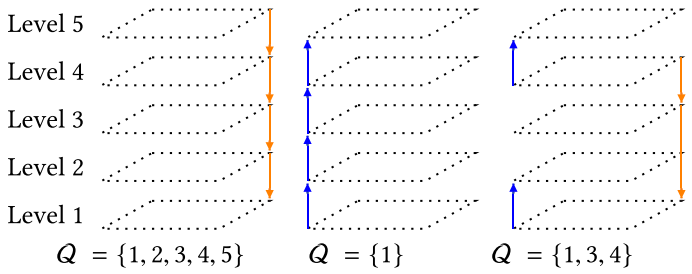


Fig. 4. Illustration of the composite heuristic for various subsets Q, with $\ell = 5$. Orange arrows pointing downward indicate propagation of edges similar to the top-down approach. Blue arrows pointing upward indicate pruning of unneeded edges, similar to the bottom-up approach.

Lemma 2.3. *For any set $Q = \{i_1, \ldots, i_m\} \subseteq \{1, 2, \ldots, \ell\}$ with $1 = i_1 < \cdots < i_m \leq \ell$, we have*

$$\mathrm{CMP}(Q) \leq \rho \sum_{k=1}^{m} (i_{k+1} - 1)\mathrm{MIN}_{i_k},$$

with the convention $i_{m+1} = \ell + 1$.

For example, $Q = \{1, 3, 4\}$ with $\ell = 5$ in Figure 4 yields $\mathrm{CMP}(Q) \leq \rho\,(2\mathrm{MIN}_1 + 3\mathrm{MIN}_3 + 5\mathrm{MIN}_4)$. With $Q = \{1, 2, 3, 4, 5\}$, we have $\mathrm{CMP}(Q) \leq \rho\,(\mathrm{MIN}_1 + 2\mathrm{MIN}_2 + \cdots + 5\mathrm{MIN}_5)$, similar to Lemma 2.2.

Proof. The proof is similar to that of Lemma 2.2: when we compute E_{i_m}, a ρ-approximate Steiner tree for T_{i_m}, we incur a cost of at most $\ell\rho\mathrm{MIN}_{i_m}$. This is due to the fact that the cost of E_{i_m} is at most $\rho\mathrm{MIN}_{i_m}$, and these edges are propagated to all levels 1 through ℓ, incurring a cost of at most $\ell\rho\mathrm{MIN}_{i_m}$. When we compute E_{i_k}, we incur a cost of at most $\rho\mathrm{MIN}_{i_k}$, and these edges are propagated to levels 1 through $i_{k+1} - 1$, incurring a cost of at most $(i_{k+1} - 1)\rho\mathrm{MIN}_{i_k}$. □

Using the trivial lower bound $\mathrm{OPT} \geq \sum_{i=1}^{\ell} \mathrm{MIN}_i$, we can find an upper bound for the approximation ratio. Without loss of generality, assume $\sum_{i=1}^{\ell} \mathrm{MIN}_i = 1$, so that $\mathrm{OPT} \geq 1$; otherwise, the edge weights can be scaled linearly. Also, since all the equations and inequalities scale linearly in ρ, we assume $\rho = 1$. Then, for a given Q, the ratio $\frac{\mathrm{CMP}(Q)}{\mathrm{OPT}}$ is upper bounded by

$$\frac{\mathrm{CMP}(Q)}{\mathrm{OPT}} \leq \frac{\rho \sum_{k=1}^{m} (i_{k+1} - 1)\mathrm{MIN}_{i_k}}{\sum_{i=1}^{\ell} \mathrm{MIN}_i} = \sum_{k=1}^{m} (i_{k+1} - 1)\mathrm{MIN}_{i_k}.$$

Given any Q, we determine an approximation ratio to the MLST problem, in a way similar to the top-down and bottom-up approaches. We start with the following lemma:

Lemma 2.4. *Let $c_1, c_2, \ldots, c_\ell$ be given non-negative real numbers with the property that $c_1 > 0$, and the nonzero ones are strictly increasing (i.e., if $i < j$ and $c_i, c_j \neq 0$, then $0 < c_i < c_j$.) Consider the following linear program: $\max c_1 y_1 + c_2 y_2 + \cdots + c_\ell y_\ell$ subject to $y_1 \geq y_2 \geq \cdots \geq y_\ell \geq 0$, and $\sum_{i=1}^{\ell} y_i = 1$. Then the optimal solution has $y_1 = y_2 = \cdots = y_k = \frac{1}{k}$ for some k, and $y_i = 0$ for $i > k$.*

Proof. Suppose that in the optimal solution, there exists some i such that $y_i > y_{i+1} > 0$. If $c_{i+1} = 0$, then setting $y_1 := y_1 + y_{i+1}$ and $y_{i+1} = 0$ improves the objective function, a contradiction. If $c_{i+1} \neq 0$, then $c_i < c_{i+1}$, and it can be shown with elementary algebra that replacing y_i and y_{i+1} by their arithmetic mean, $\frac{y_i + y_{i+1}}{2}$, improves the objective function as well. □

A simple corollary of Lemma 2.4 is that the maximum value of the objective in the LP equals $\max\left(c_1, \frac{1}{2}(c_1+c_2), \frac{1}{3}(c_1+c_2+c_3), \ldots, \frac{1}{\ell}(c_1+\cdots+c_\ell)\right)$. For a given Q (assuming $\rho = 1$), the composite heuristic on $Q = \{i_1, \ldots, i_m\}$ is a t-approximation, where t is the solution to the following simple linear program: max t subject to $t \le \sum_{k=1}^{m}(i_{k+1}-1)y_{i_k}$; $y_1 \ge y_2 \ge \cdots \ge y_\ell \ge 0$; $\sum_{i=1}^{\ell} y_i = 1$. As this LP is of the form given in Lemma 2.4, we can easily compute an approximation ratio for a given Q as

$$t(Q) = \max_{m' \le m} \frac{\sum_{k=1}^{m'}(i_{k+1}-1)}{i_{m'}}.$$

For example, the corresponding objective function for TOP is $\max y_1 + 2y_2 + \cdots + \ell y_\ell$, and the maximum equals $\max(1, \frac{1}{2}(1+2), \ldots, \frac{1}{\ell}(1+2+\cdots+\ell)) = \frac{\ell+1}{2}$, which is consistent with Theorem 2.1. The corresponding objective function for BOT is $\max \ell = \ell$.

An important choice of Q is $Q = \{1, 2, 4, \ldots, 2^m\}$ where $m = \lfloor \log_2 \ell \rfloor$. Charikar et al. [7] show that this is a 4-approximation, assuming $\rho = 1$. Indeed, according to the above formula $t = \max(1, \frac{(2-1)+(2^2-1)}{2^1}, \ldots, \frac{\sum_{i=0}^{m}(2^{i+1}-1)}{2^m}) = 4 - m/2^m \le 4$.

When $\ell \ge 2$, there are $2^{\ell-1}$ possible subsets Q, giving $2^{\ell-1}$ possible heuristics. In particular, for $\ell = 2$, the only $2^{2-1} = 2$ heuristics are top-down and bottom-up (Section 2.1). The composite algorithm executes all possible heuristics and selects the MLST with minimum cost (denoted CMP):

$$\mathrm{CMP} = \min_{\substack{Q \subseteq \{1,\ldots,\ell\} \\ 1 \in Q}} \mathrm{CMP}(Q).$$

Theorem 2.5. *For $\ell \ge 2$, the composite heuristic produces a t_ℓ-approximation, where t_ℓ is the solution to the following linear program (LP):*

$$\begin{aligned}
&\max t \\
&\textit{subject to} \quad t \le \sum_{k=1}^{m}(i_{k+1}-1)y_{i_k} && \forall\, Q = \{i_1, \ldots, i_m\} \\
&\qquad y_i \ge y_{i+1} && \forall\, 1 \le i \le \ell - 1 \\
&\qquad \sum_{i=1}^{\ell} y_i = 1 \\
&\qquad y_i \ge 0 && \forall\, 1 \le i \le \ell.
\end{aligned}$$

Proof. Again, we assume, without loss of generality, that $\rho = 1$ and that $\sum_{i=1}^{\ell} \mathrm{MIN}_i = 1$. Given an instance of MLST and the corresponding values $\mathrm{MIN}_1, \ldots, \mathrm{MIN}_\ell$, let $Q^* = \{i_1, \ldots, i_m\}$ denote the subset of $\{1, \ldots, \ell\}$ for which the quantity $\sum_{k=1}^{m}(i_{k+1}-1)\mathrm{MIN}_{i_k}$ is minimized. Then by Lemma 2.3, we have $\mathrm{CMP} \le \mathrm{CMP}(Q^*) \le \sum_{k=1}^{m}(i_{k+1}-1)\mathrm{MIN}_{i_k} = \hat{t}$. So, for a specific instance of the MLST problem, CMP is upper bounded by $\hat{t}$, which is the minimum over $2^{\ell-1}$ different expressions, all linear combinations of $\mathrm{MIN}_1, \ldots, \mathrm{MIN}_\ell$.

As t_ℓ is the maximum of the objective over all feasible $\mathrm{MIN}_1, \ldots, \mathrm{MIN}_\ell$, we have $\hat{t} \le t_\ell$, so $\mathrm{CMP} \le t_\ell = t_\ell \cdot \mathrm{OPT}$ as desired. □

The above LP has $\ell + 1$ variables and $2^{\ell-1} + 2\ell$ constraints. Each subset $Q \in \{1, 2, \ldots, \ell\}$ with $1 \in Q$ corresponds to one constraint.

Lemma 2.6. *The system of $2^{\ell-1}$ inequalities can be expressed in matrix form as*

$$t \cdot \mathbf{1}_{2^{\ell-1}\times 1} \le M_\ell \boldsymbol{y},$$

where $\boldsymbol{y} = [y_1, y_2, \ldots, y_\ell]^T$ *and* M_ℓ *is a* $(2^{\ell-1} \times \ell)$*-matrix that can be constructed recursively as*

$$M_\ell = \begin{bmatrix} P_{\ell-1} + M_{\ell-1} & \mathbf{0}_{2^{\ell-2}\times 1} \\ M_{\ell-1} & \ell \cdot \mathbf{1}_{2^{\ell-2}\times 1} \end{bmatrix} \text{ and } P_\ell = \begin{bmatrix} P_{\ell-1} & \mathbf{0}_{2^{\ell-2}\times 1} \\ \mathbf{0}_{2^{\ell-2}\times(\ell-1)} & \mathbf{1}_{2^{\ell-2}\times 1} \end{bmatrix},$$

starting with the 1×1 *matrices* $M_1 = [1]$ *and* $P_1 = [1]$.

Proof. The idea is that the rows of P_ℓ encode the largest element of their corresponding subsets—if we list the $2^{\ell-1}$ subsets of $\{1, 2, \ldots, \ell\}$ in the usual ordering ($\{1\}$, $\{1, 2\}$, $\{1, 3\}$, $\{1, 2, 3\}$, $\{1, 4\}$, ...), then $P_{i,j} = 1$ if j is the largest element of the ith subset.

We provide proof by induction. If $\ell = 1$, then we have only one possible subset of levels $\boldsymbol{Q} = \{1\}$ and one inequality: $t < y_1$. Hence, $M_1 = [1]$ and $P_1 = [1]$. We assume that the claim is true for $\ell - 1$ levels. To show that the claim is also true for ℓ levels, we first prove that the recursive construction of P_ℓ is valid. We now have one more level, which is the ℓth level. Hence, all possible subsets of levels for $\ell - 1$ levels are still valid subsets of levels for the new configuration, where we have one additional level. However, we now have $\ell - 1$ additional subsets of levels, each of which can be generated from a previous subset of levels by just adding the new level ℓ in the subset. The previous subset of levels will have the same encoding in P_ℓ. Hence, we just concatenate $\mathbf{0}_{2^{\ell-2}\times 1}$ in the upper right corner of of P_ℓ. For the new possible subsets, the largest element is always ℓ. Which validates the recursive formulation of P_ℓ.

It remains to show that the recursive construction of M_ℓ is valid. Given $M_{\ell-1}$ and $P_{\ell-1}$, we can construct M_ℓ by casework on whether $\ell \in \boldsymbol{Q}$ or not. If $\ell \notin \boldsymbol{Q}$, then we build the first $2^{\ell-2}$ rows by using the previous matrix $M_{\ell-1}$, and adding 1 to the rightmost nonzero entry of each column (which is equivalent to adding $P_{\ell-1}$). If $\ell \in \boldsymbol{Q}$, then we build the remaining $2^{\ell-2}$ rows by using the previous matrix $M_{\ell-1}$, and appending a $2^{\ell-2} \times 1$ column whose values are equal to ℓ. □

This recursively defined matrix is not central to the composite algorithm, but gives a nice way of formulating the LP above.

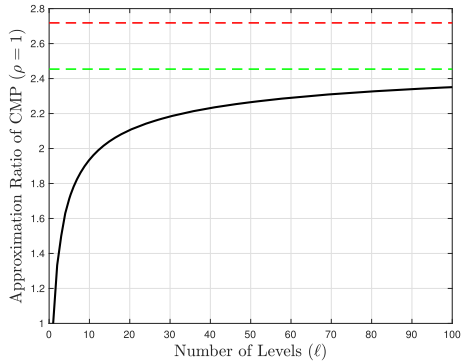
Solving the above LP directly is challenging for large ℓ due to its size. We instead use a column generation method. The idea is that solving the LP for only a subset of the constraints will produce an upper bound for the approximation ratio—larger values can be returned due to relaxing the constraints. Now, the objective would be to add "effective" constraints that would be most needed for getting a more accurate solution.

In our column generation, we only add one single constraint at a time. Let $\boldsymbol{Q}$ denote the set of all the constraints at the running step. Solving the LP provides a vector $\mathbf{y}$ and an upper bound for the approximation ratio t. Our goal is to find a new set $\boldsymbol{Q}_{\text{new}} = \{i_1, i_2, \ldots, i_k\}$ that gives the smallest value of $\sum_{k=1}^{m}(i_{k+1} - 1)y_{i_k}$ given the vector $\mathbf{y}$ from the current LP solver. We can use an ILP to find the set $\boldsymbol{Q}_{\text{new}}$. Specifically, we define indicator variables θ_{ij} so that $\theta_{ij} = 1$ if and only if i and j are consecutive level choices for the new constraint $\boldsymbol{Q}_{\text{new}}$, and $\theta_{ij} = 0$, otherwise. For example, for $\boldsymbol{Q}_{\text{new}} = \{1, 3, 7\}$ with $\ell = 10$, we must have $\theta_{1,3} = \theta_{3,7} = \theta_{7,11} = 1$ and all other θ_{ij}'s equal to zero.

Given a vector $\mathbf{y} = [y_1, \ldots, y_\ell]$, the choice of $\boldsymbol{Q}_{\text{new}} = \{i : \theta_{ij} = 1, i < j \text{ and } i, j \in \{1, 2, \ldots, \ell\}\}$.

Lemma 2.7. *The following ILP minimizes* $\sum_{k=1}^{m}(i_{k+1} - 1)y_{i_k}$*, where* i_k *is the* kth *smallest element of* $\boldsymbol{Q}$ *and* $i_{m+1} = \ell + 1$.

$$\begin{aligned} &\textit{Minimize} && \sum_{i=1}^{\ell}\sum_{j=i+1}^{\ell+1}(j-1)\theta_{ij}y_i && \\ &\textit{subject to} && \sum_{i:j>i}\theta_{ij} \le 1, && \forall j \in \{2, \ldots, \ell+1\} \end{aligned}$$



ℓ	t_ℓ	ℓ	t_ℓ
1	ρ	12	1.986ρ
2	1.333ρ	13	2.007ρ
3	1.500ρ	14	2.025ρ
4	1.630ρ	15	2.041ρ
5	1.713ρ	16	2.056ρ
6	1.778ρ	17	2.070ρ
7	1.828ρ	18	2.083ρ
8	1.869ρ	19	2.094ρ
9	1.905ρ	20	2.106ρ
10	1.936ρ	50	2.265ρ
11	1.963ρ	100	2.351ρ

Fig. 5. Approximation ratios for the composite algorithm for $\ell = 1, \ldots, 100$ (black curve), compared to the ratio $t = e\rho$ (red dashed line) guaranteed by the algorithm of Charikar et al. [7] and $t = 2.454\rho$ (green dashed line) guaranteed by the algorithm of Karpinski et al. [14]. The table to the right lists the exact values of t_ℓ.

$$\sum_{j:i<j} \theta_{ij} \leq 1, \qquad \forall i \in \{1, \ldots, \ell\}$$

$$\sum_{i:i<k} \theta_{ik} = \sum_{j:j>k} \theta_{kj} \qquad \forall k \in \{2, \ldots, \ell\}$$

$$\sum_{j:1<j} \theta_{1j} = 1,$$

$$\sum_{i:i<\ell+1} \theta_{i(\ell+1)} = 1,$$

$$\theta_{ij} \in \{0, 1\}.$$

Proof. Using the indicator variables, $\sum_{k=1}^{m}(i_{k+1} - 1)y_{i_k}$ can be expressed as $\sum_{i=1}^{\ell}\sum_{j=i+1}^{\ell+1}(j - 1)\theta_{ij}y_i$, because $\theta_{i_k,i_{k+1}} = 1$ and the other θ_{ij}'s are zero. In the above formulation, the first constraint indicates that, for every given i or j, at most one θ_{ij} is equal to one. The second constraint indicates that, for a given k, if $\theta_{ik} = 1$ for some i, then there is also a j such that $\theta_{kj} = 1$. In other words, there are no overlapping intervals. For example, $\theta_{3,7}$ and $\theta_{4,9}$ cannot both equal 1, because $[3, 7]$ overlaps $[4, 9]$. Consider the example provided above; $Q_{\text{new}} = \{1, 3, 7\}$. Here, $\theta_{1,3} = 1$. Hence, there is exactly one possible $j > 3$ such that $\theta_{3,j} = 1$. In our example, this is $j = 7$. The last constraints guarantees that level 1 is chosen for Q_{new}. □

Lemma 2.7 allows for a column generation technique to solve the LP of Theorem 2.5 for computing the approximation ratio of CMP algorithm. We initially start with an empty set Q and $\mathbf{y} = [1, \frac{1}{2}, \ldots, \frac{1}{\ell}]$. Using Lemma 2.7, we then find Q_{new} and add it to the constraint set Q. We repeat the process until Q_{new} already belongs to Q. Without the column generation technique, we could reach only $\ell = 22$ levels. The new techniques, however, allows us to solve for much larger values of ℓ. In the following, we report the results up to 100 levels.

Theorem 2.8. *For any $\ell = 2, \ldots, 100$, the MLST returned by the composite algorithm yields a solution whose cost is at most $t_\ell \cdot$ OPT, where the values of t_ℓ are listed in Figure 5.*

Neglecting the factor ρ for now (i.e., assuming $\rho = 1$), the approximation ratio $t = 3/2$ for $\ell = 3$ is slightly better than the ratio of $(5 + 4\sqrt{2})/7 + \varepsilon \approx 1.522 + \varepsilon$ guaranteed by Xue et al. [23] for the Euclidean case. (The additive constant ε in their ratio stems from using Arora's PTAS as a

subroutine for Euclidean ST, which corresponds to the multiplicative constant ρ for using an ST algorithm as a subroutine for MLST.) Recall that an improvement for $\ell = 3$ was posed as an open problem by Karpinski et al. [14]. Also, for each of the cases $4 \leq \ell \leq 100$ our results in Theorem 2.8 improve the approximation ratios of $e\rho \approx 2.718\rho$ and 2.454ρ guaranteed by Charikar et al. [7] and by Karpinski et al. [14], respectively. The graph of the approximation ratio of the composite algorithm (see Figure 5) for $\ell = 1, \ldots, 100$ suggests that it will stay below 2.454ρ for values of ℓ much larger than 100.

The obvious disadvantage is that computing CMP involves $2^{\ell-1}$ different heuristics, requiring $2^{\ell-2}(\ell+1)$ ST computations, which is not feasible for large ℓ. In the following, we show that we can achieve the same approximation guarantee with at most $O(\ell)$ ST computations.

THEOREM 2.9. *Given an instance of the MLST problem, a specific subset $\mathcal{Q}^* \subseteq \{1, 2, \ldots, \ell\}$ (with $1 \in \mathcal{Q}^*$) can be found through ℓ ST computations, such that running the composite heuristic on $\mathcal{Q}^*$ is a t_ℓ-approximation.*

PROOF. Given a graph $G = (V, E)$ with cost function c, and terminal sets $T_\ell \subset T_{\ell-1} \subset \cdots \subset T_1 \subseteq V$, compute a Steiner tree on each level and set $\mathrm{MIN}_i = c(ST(G, T_i))$. Again, assume w.l.o.g. that $\sum_{i=1}^{\ell} \mathrm{MIN}_i = 1$, which can be done by scaling the edge weights appropriately after computing the minimum Steiner trees.

Since $\boldsymbol{y} = [\mathrm{MIN}_1, \ldots, \mathrm{MIN}_\ell]^T$ and $t = \min M_\ell \boldsymbol{y}$ is a feasible, but not necessarily optimal, solution to the LP for computing the approximation ratio t_ℓ, we have $\min M_\ell \boldsymbol{y} = t \leq t_\ell$. Let $q \in \{1, 2, \ldots, 2^{\ell-1}\}$ be the row of M_ℓ whose dot product with $\boldsymbol{y}$ is minimized (i.e., equals t), and let $\mathcal{Q}^*$ be the subset of levels corresponding to the qth row of M_ℓ, which can be obtained from the non-zero entries of the qth row. Then for this specific subset $\mathcal{Q}^*$ of levels, we have $\mathrm{CMP}(\mathcal{Q}^*) \leq t \leq t_\ell \leq t_\ell \mathrm{OPT}$. The optimal choice of $\mathcal{Q}^*$ given $\boldsymbol{y} = [\mathrm{MIN}_1, \ldots, \mathrm{MIN}_\ell]^T$ can be obtained very efficiently using the ILP with column generation. □

Note that the total number of ST computations is reduced from $2^{\ell-2}(\ell+1)$ to at most 2ℓ; we need ℓ ST computations to determine $\mathcal{Q}_{\mathrm{new}}$ (according to the proof of Theorem 2.9) and another at most ℓ ST computations to "execute" $\mathcal{Q}_{\mathrm{new}}$. The resulting solution with cost $\mathrm{CMP}(\mathcal{Q}^*)$ does not necessarily have the same cost as the solution with cost CMP, however, the solution returned is still at most t_ℓ times the optimum. It is worth noting that the analyses in this section did not assume that the computed edge sets are trees, only that the edge sets are nested and that the cost of a solution is the sum of the costs over all levels.

3 INTEGER LINEAR PROGRAMMING (ILP) FORMULATIONS

In this section, we discuss several different ILP formulations for the MLST problem.

3.1 ILP Based on Cuts

This is a standard ILP formulation for the (single-level) Steiner tree problem. Recall that $T \subseteq V$ is the set of terminals. Given a cut $S \subseteq V$, let $\delta(S) = \{uv \in E \mid u \in S, v \notin S\}$ denote the set of all edges that have exactly one endpoint in S. Given an undirected edge $uv \in E$, let $x_{uv} = 1$ if uv is present in the solution, 0 otherwise. An ILP formulation for ST is as follows:

$$\text{Minimize} \quad \sum_{uv \in E} c(u, v) \cdot x_{uv}$$

$$\text{subject to} \quad \sum_{uv \in \delta(S)} x_{uv} \geq 1 \qquad \forall\, S \subset V; S \cap T \neq \emptyset; S \cap T \neq T$$

$$x_{uv} \in \{0, 1\} \qquad \forall uv \in E.$$

The cut-based formulation generalizes to ℓ levels naturally. Let $x^i_{uv} = 1$ if edge uv is present on level i, and 0 otherwise. We constrain that the graph on level i is a subgraph of the graph on level $i-1$ by requiring $x^i_{uv} \le x^{i-1}_{uv}$ for all $2 \le i \le \ell$ and for all $uv \in E$. Then a cut-based formulation for the MLST problem is as follows:

$$\begin{aligned}
\text{Minimize} \quad & \sum_{i=1}^{\ell} \sum_{uv \in E} c(u,v) \cdot x^i_{uv} \\
\text{subject to} \quad & \sum_{uv \in \delta(S)} x^i_{uv} \ge 1 && \forall\, S \in V; S \cap T \neq \emptyset, T; 1 \le i \le \ell \\
& x^i_{uv} \le x^{i-1}_{uv} && \forall uv \in E; 2 \le i \le \ell \\
& x^i_{uv} \in \{0,1\} && \forall uv \in E; 1 \le i \le \ell.
\end{aligned}$$

The number of variables is $O(\ell \cdot |E|)$, however, the number of constraints is $O(\ell \cdot 2^{|V|})$.

3.2 ILP Based on Multi-commodity Flow

We recall here the well-known undirected flow formulation for ST [2, 18]. Let $s \in T$ be a fixed terminal node, the *source*. Given an edge $uv \in E$, the indicator variable x_{uv} equals 1 if the edge uv is present in the solution and 0 otherwise. This formulation sends $|T|-1$ unit commodities from the source s to each terminal in $T - \{s\}$. The variable f^p_{uv} denotes the (integer) flow from u to v of commodity p. A multi-commodity ILP formulation for ST is as follows:

$$\begin{aligned}
\text{Minimize} \quad & \sum_{uv \in E} c(u,v) \cdot x_{uv} \\
\text{subject to} \quad & \sum_{vw \in E} f^p_{vw} - \sum_{uv \in E} f^p_{uv} = \begin{cases} 1 & \text{if } v = s \\ -1 & \text{if } v = p \\ 0 & \text{otherwise} \end{cases} && \forall\, v \in V \\
& 0 \le f^p_{uv} \le 1 && \forall\, uv \in E \\
& 0 \le f^p_{vu} \le 1 && \forall\, uv \in E \\
& x_{uv} \in \{0,1\} && \forall\, uv \in E.
\end{aligned}$$

To generalize to the MLST problem, we add the linking constraints ($x^i_{uv} \le x^{i-1}_{uv}$ as before) and enforce the flow constraints on levels $1, \ldots, \ell$.

3.3 ILP Based on Single-commodity Flow

ST can also be formulated using a single-commodity flow, instead of multiple commodities. Here, we will assume the input graph is directed, by replacing each edge uv with directed edges (u,v) and (v,u) of the same cost. As before, f_{uv} denotes the flow from u to v:

$$\begin{aligned}
\text{Minimize} \quad & \sum_{(u,v) \in E} c(u,v) \cdot x_{uv} \\
\text{subject to} \quad & \sum_{(v,w) \in E} f_{vw} - \sum_{(u,v) \in E} f_{uv} = \begin{cases} |T|-1 & \text{if } v = s \\ -1 & \text{if } v \in T \setminus \{s\} \\ 0 & \text{otherwise} \end{cases} && \forall\, v \in V \\
& 0 \le f_{uv} \le (|T|-1) \cdot x_{uv} && \forall\, (u,v) \in E \\
& x_{uv} \in \{0,1\} && \forall\, (u,v) \in E.
\end{aligned}$$

To generalize to multiple levels, we add the linking constraints $x^i_{uv} \geq x^{i-1}_{uv}$ as before. Let f^i_{uv} denote the integer flow along edge (u, v) on level i. Let $s \in T_\ell$ be a source terminal on the top level T_ℓ. Then the MLST problem can be formulated using single-commodity flows on each level:

$$\text{Minimize} \quad \sum_{i=1}^{\ell} \sum_{(u,v)\in E} c(u,v) x^i_{uv}$$

$$\text{subject to} \sum_{(v,w)\in E} f^i_{vw} - \sum_{(u,v)\in E} f^i_{uv} = \begin{cases} |T_i| - 1 & \text{if } v = s \\ -1 & \text{if } v \in T_i \setminus \{s\} \\ 0 & \text{else} \end{cases} \qquad \forall\, v \in V; 1 \leq i \leq \ell$$

$$x^i_{uv} \geq x^{i-1}_{uv} \qquad \forall\, (u,v) \in E; 2 \leq i \leq \ell$$

$$0 \leq f^i_{uv} \leq (|T_i| - 1) \cdot x^i_{uv} \qquad \forall\, (u,v) \in E; 1 \leq i \leq \ell$$

$$x^i_{uv} \in \{0, 1\} \qquad \forall\, (u,v) \in E; 1 \leq i \leq \ell.$$

The number of variables is $O(\ell|E|)$ and the number of constraints is $O(\ell(|E| + |V|))$. In Section 3.4, we reduce the number of variables and constraints to $O(|E|)$ and $O(|E| + |V|)$, respectively.

3.4 A Smaller ILP Based on Single-commodity Flow

We can simplify the flow-based ILP in Section 3.3 so that the number of variables is $O(|E|)$ and the number of constraints is $O(|E| + |V|)$. This is done by only enforcing the single-commodity flow on the bottom level. Let $L(v)$ denote the highest level that v is a terminal in, i.e., if $v \in T_i$ and $v \notin T_{i+1}$, then $L(v) = i$. If $v \notin T_1$, then $L(v) = 0$. For each directed edge $(u, v) \in E$, let $x_{uv} = 1$ if (u, v) appears on the bottom level (level 1) in the solution, and $x_{uv} = 0$ otherwise. Let y_{uv} denote the highest level that (u, v) appears in, i.e., $y_{uv} = i$ if (u, v) is present on level i but not on level $i + 1$, and $y_{uv} = 0$ if (u, v) is not present anywhere. The variables y_{uv} indicate the number of times we pay the cost of edge (u, v) in the solution. Let $N(v) = \{u \in V \mid (u, v) \in E\}$ denote the neighborhood of v. As in Section 3.3, let f_{uv} denote the flow along directed edge (u, v), and let $s \in T_\ell$ be the source. A reduced ILP formulation is as follows:

$$\text{Minimize} \quad \sum_{uv\in E} c(u,v)(y_{uv} + y_{vu}), \tag{3.1}$$

$$\text{subject to} \sum_{(v,w)\in E} f_{vw} - \sum_{(u,v)\in E} f_{uv} = \begin{cases} |T_1| - 1 & v = s \\ -1 & v \in T_1 - \{s\} \\ 0 & \text{otherwise} \end{cases} \qquad \forall v \in V, \tag{3.2}$$

$$0 \leq f_{uv} \leq (|T_1| - 1) x_{uv} \qquad \forall uv \in E, \tag{3.3}$$

$$\sum_{u\in N(v)} x_{uv} \leq 1 \qquad \forall v \in V, \tag{3.4}$$

$$x_{uv} \leq y_{uv} \leq \ell x_{uv} \qquad \forall uv \in E, \tag{3.5}$$

$$\sum_{u\in N(v)-\{w\}} y_{uv} \geq y_{vw} \qquad \forall vw \in E, v \neq s, \tag{3.6}$$

$$\sum_{u\in N(v)} y_{uv} \geq L(v) \qquad \forall v \in T_1 - \{s\}, \tag{3.7}$$

$$x_{uv} \in \{0, 1\} \qquad \forall uv \in E. \tag{3.8}$$

Constraint Equations (3.2) and (3.3) are the same as before, but we only enforce the flow constraint on the bottom level (level 1).

Constraint Equation (3.4) enforces that each vertex has at most one incoming edge in the solution.

The combination of constraint Equations (3.2) and (3.3) ensures that if (v, w) is an edge in the solution, and v is not the root, then there is some $u \neq w$ for which (u, v) is an edge in the solution. Combined with constraint Equation (3.4), this implies that v has exactly one incoming edge, and it is not the edge (w, v).

Constraint Equation (3.5) ensures that if $x_{uv} = 0$, then we do not incur cost for edge (u, v), and so $y_{uv} = 0$. Otherwise, if $x_{uv} = 1$, then we pay for edge (u, v) between 1 and ℓ times, or $1 \leq y_{uv} \leq \ell$.

Constraint Equation (3.6) ensures that if edge (v, w) appears on levels $1, \ldots, i$ (i.e., $y_{uv} = i$), and v is not the root, then the sum over all neighbors u of v (other than w) of y_{uv} is at least i. As v has exactly one incoming edge (u, v) and $u \neq w$, constraint Equation (3.6) combined with constraint Equations (3.4) and (3.5) together imply that v has exactly one incoming edge, and its level is greater than or equal to y_{vw}.

Constraint Equation (3.7) ensures that if v is any terminal besides the root, then the sum over all neighbors u of y_{uv} is at least its level, $L(v)$. Combined with constraint Equations (3.4) and (3.5), this implies that every non-root terminal has exactly one incoming edge that appears on at least $L(v)$ levels.

Constraint Equation (3.8) ensures that all x_{uv} variables are binary, which implies $0 \leq y_{uv} \leq \ell$ for all variables y_{uv}. Given a solution to the above ILP, the graph G_1 is such that $uv \in G_1$ if $x_{uv} = 1$ or $x_{vu} = 1$. More generally, $uv \in G_i$ if $y_{uv} \geq i$ or $y_{vu} \geq i$.

Lemma 3.1. *In the* optimal *solution to the above ILP, the graph* $G_1 = (V, E_1)$ *with* $E_1 = \{uv \in E \mid x_{uv} = 1$ *or* $x_{vu} = 1\}$ *is a Steiner tree spanning the terminals in* T_1.

Proof. We show that (i) G_1 contains no cycle, (ii) there exists a path in G_1 from the source s to every terminal $v \in T_1$, and (iii) G_1 is connected.

(i) Assume otherwise that G_1 contains a cycle C. Such a cycle contains directed edges oriented in the same direction; otherwise, there exist vertices u, v, w along the cycle such that $x_{uv} = x_{wv} = 1$, which violates Equation (3.4). Additionally, such a cycle cannot have any "incoming" edges (edges (u, v) where $u \notin C$ and $v \in C$), as this violates Equation (3.4) as well.

If C contains s, then removing its preceding edge (v, s) and reducing all flows on C by f_{vs} yields a feasible solution with lower cost, contradicting optimality. If C does not contain s, but contains some terminal $v \in T_1$, then since there are no incoming edges into C, we cannot satisfy the flow constraint on v. If C contains no terminal, then removing C along with its edge flows gives a solution with lower cost.

(ii) Consider an arbitrary terminal $v \in T_1$ with $v \neq s$. Then using previous arguments, v has exactly one incoming edge (u, v) (whose level is at least $L(v)$). If $u = s$, we are done. Otherwise, we continue this process until we reach the source s. Note that continuing this process does not revisit a vertex, as G_1 contains no cycle.

(iii) Since there exists a directed path from s to each terminal $v \in T_1$, then all terminals are in the same connected component in G_1. If there exist other connected components in G_1, then removing them and setting flows to zero yields a solution with lower cost. □

Lemma 3.2. *In the* optimal *solution to the above ILP, the graph* $G_i = (V, E_i)$ *with* $E_i = \{uv \in E \mid y_{uv} \geq i$ *or* $y_{vu} \geq i\}$ *is a Steiner tree spanning all terminals* T_i.

Proof. The graph G_i is necessarily a subgraph of G_1, since $y_{uv} \geq i$ or $y_{vu} \geq i$ implies $x_{uv} \geq 1$ or $x_{vu} \geq 1$ by Equation (3.5). Consider some terminal $v \in T_i$, $v \neq s$. By constraint Equation (3.7),

there is exactly one incoming edge (u, v) to v such that $y_{uv} \geq L(v)$. Applying similar arguments using constraint (3.6), we will eventually reach the root via a path, all of whose edges appear at least $L(v)$ times. □

Theorem 3.3. *The optimal solution to the above ILP with cost* OPT_{ILP} *yields the optimal MLST solution.*

Proof. The optimal MLST solution whose cost is OPT is a feasible solution to the ILP, as we can set the x_{uv}, y_{uv}, and f_{uv} variables accordingly, so $\text{OPT}_{ILP} \leq \text{OPT}$. By Lemmas 3.1 and 3.2, the optimal solution OPT_{ILP} gives a feasible solution to the MLST problem, so $\text{OPT} \leq \text{OPT}_{ILP}$. Then $\text{OPT} = \text{OPT}_{ILP}$. □

The number of flow variables is $2|E|$ (where $|E|$ is the number of edges in the input graph), and the total number of variables is $O(|E|)$. The number of flow constraints is $O(|V|)$ and the total number of constraints is $O(|V| + |E|)$. Additionally, the integrality constraints on the flow variables f_{uv} as well as the variables y_{uv} may be dropped without affecting the optimal solution.

4 EXPERIMENTAL RESULTS

Graph Data Synthesis. The graphs we used in our experiment are synthesized from graph generation models. In particular, we used three random network generation models: Erdős–Renyi (ER) [10], Watts–Strogatz (WS) [21], and Barabási–Albert (BA) [3]. These networks are very well studied in the literature [17].

The Erdős–Renyi model, $\text{ER}(n, p)$, assigns an edge to every possible pair among $n = |V|$ vertices with probability p, independently of other edges. It is well-known that an instance of $\text{ER}(n, p)$ with $p = (1 + \varepsilon)\frac{\ln n}{n}$ is almost surely connected for $\varepsilon > 0$ [10]. For our experiment, $n = 50, 100, 150, \ldots, 500$, and $\varepsilon = 1$.

The Watts–Strogatz model, $\text{WS}(n, K, \beta)$, initially creates a ring lattice of constant degree K, and then rewires each edge with probability $0 \leq \beta \leq 1$ while avoiding self-loops or duplicate edges. Interestingly, the Watts–Strogatz model generates graphs that have the small-world property while having high clustering coefficient [21]. In our experiment, the values of K and β are equal to 6 and 0.2, respectively.

The Barabási–Albert model, $\text{BA}(m_0, m)$, uses a preferential attachment mechanism to generate a growing scale-free network. The model starts with a graph of m_0 vertices. Then, each new vertex connects to $m \leq m_0$ existing nodes with probability proportional to its instantaneous degree. The BA model generates networks with power-law degree distribution, i.e., few vertices become hubs with extremely large degree [3]. This model is a network growth model. In our experiments, we let the network grow until a desired network size n is attained. We vary m_0 from 10 to 100 in our experiments. We keep the value of m equal to 5.

For each generation model, we generate graphs on size $|V| = 50, 100, 150, \ldots, 500$. On each graph instance, we assign integer edge weights $c(e)$ randomly and uniformly between 1 and 10 inclusive. We only consider connected graphs in our experiment. Computational challenges of solving an ILP limit the size of the graphs to a few hundred in practice.

Selection of Levels and Terminals. For each generated graph, we generated MLST instances with $\ell = 2, 3, 4, 5$ levels. We adopted two strategies for selecting the terminals on the ℓ levels: *linear* vs. *exponential*. In the linear case, we select the terminals on each level by randomly sampling $\lfloor |V| \cdot (\ell - i + 1)/(\ell + 1) \rfloor$ vertices on level i so that the size of the terminal sets decreases linearly. As the terminal sets are nested, T_i can be selected by sampling from T_{i-1} (or from V if $i = 1$). In the exponential case, we select the terminals on each level by randomly sampling $\lfloor |V|/2^{\ell - i + 1} \rfloor$ vertices so that the size of the terminal sets decreases exponentially by a factor of 2.

To summarize, a single instance of an input to the MLST problem is characterized by four parameters: network generation model NGM ∈ {ER,WS,BA}, number of vertices $|V|$, number of levels ℓ, and the terminal selection method TSM ∈ {LINEAR,EXPONENTIAL}. Since each instance of the experiment setup involves randomness at different steps, we generated five instances for every choice of parameters (e.g., WS, $|V| = 100$, $\ell = 5$, LINEAR).

Algorithms and Outputs. We implemented the bottom-up, top-down, and composite heuristics described in Section 2.

For evaluating the heuristics, we also implemented all ILPs described in Section 3 using CPLEX 12.6.2 as an ILP solver. The ILP described in Section 3.4 works very well in practice. Hence, we have used this ILP for our experiment. The model of the HPC system we used for our experiment is Lenovo NeXtScale nx360 M5. It is a distributed system; the models of the processors in this HPC are Xeon Haswell E5-2695 Dual 14-core and Xeon Broadwell E5-2695 Dual 14-core. The speed of a processor is 2.3 GHz. There are 400 nodes each having 28 cores. Each node has 192 GB memory. The operating system is CentOS 6.10.

For each instance of the MLST problem, we compute the costs of the MLST from the ILP solution (OPT), the bottom-up solution (BOT), the top-down solution (TOP), the quality-of-service solution (QoS) from [7], the composite heuristic (CMP), and the guaranteed performance heuristic (CMP(Q^*) where Q^* is chosen suitably). The four heuristics involve a (single-level) ST subroutine; we used both the 2-approximation algorithm of Gilbert and Pollak [11], as well as the flow formulation described in Section 3.4, which solves ST optimally. The purpose of this is to assess whether solving ST optimally significantly improves the approximation ratio.

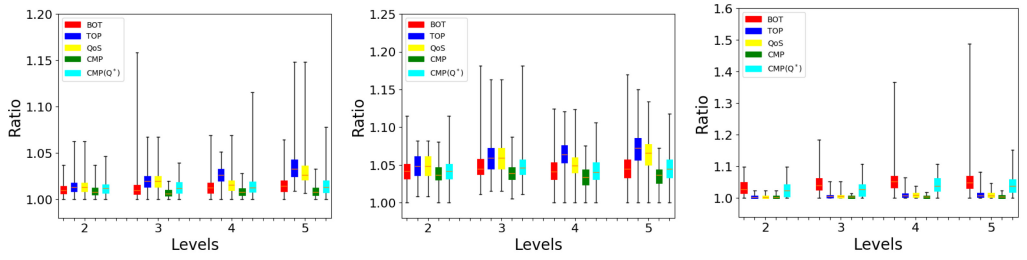
After completing the experiment, we compared the results of the heuristics with exact solutions. We show the performance ratio for each heuristic (defined as the heuristic cost divided by OPT), and how the ratio depends on the experiment parameters (number of levels ℓ, terminal selection method, number of vertices $|V|$). We record the number of ST computations involved for the guaranteed performance heuristic (CMP(Q^*)) (note that this equals $\ell + |Q^*|$). Finally, we discuss the running time of the ILP we have used in our experiment. All box plots shown below show the minimum, interquartile range (IQR) and maximum, aggregated over all instances using the parameter being compared.

Results. We found that the four heuristics perform very well in practice using the 2-approximation algorithm as a (single-level) ST subroutine, and that using an exact ST subroutine did not significantly improve performance. Hence, we only discuss the results that use the 2-approximation as a subroutine.

Figure 6 shows the performance of the four heuristics compared with the optimal solution as a function of ℓ. As expected, the performance of the heuristics gets slightly worse as ℓ increases. The bottom-up approach had the worst performance, while the composite heuristic performed very well in practice.

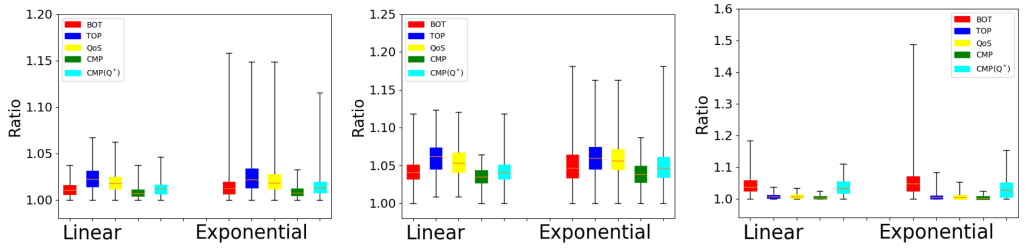
Figure 7 shows the performance of the four heuristics compared with the optimal solution as a function of terminal selection, either LINEAR or EXPONENTIAL. Overall, the heuristics performed slightly worse when the sizes of the terminal sets decrease exponentially.

Figures 8 through 10 show the performance of the heuristics compared with the optimal solution, as a function of the number of vertices $|V|$. The minimum, average, and maximum values for "Ratio" are aggregated over all instances of $|V|$ vertices (terminal selection, number of levels ℓ, 5 instances for each). Due to space, we omit the bottom-up (BU) heuristic here, which tends to be comparable or slightly worse in performance than the top-down (TD) heuristic. Again, the composite (CMP) has the best ratio as it selects the best over all $2^{\ell-1}$ possible solutions; top-down and CMP(Q^*) were comparable.



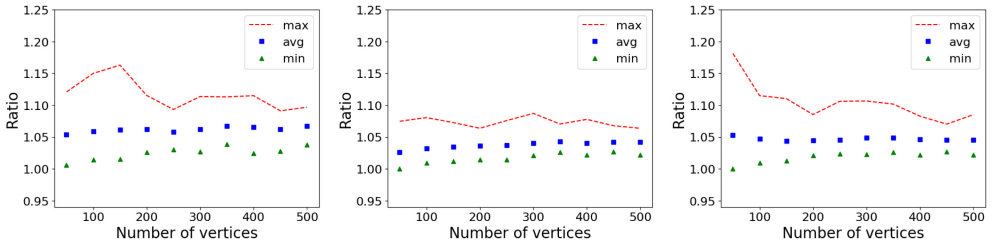
(a) Barabási–Albert (b) Erdős–Rényi (c) Watts–Strogatz

Fig. 6. Performance of BOT, TOP, QoS, CMP, and CMP(Q^*) w.r.t. the number ℓ of levels using the 2-approximation for ST as a subroutine. Note that the range of the Y-axis is not the same for all plots in the row.



(a) Barabási–Albert (b) Erdős–Rényi (c) Watts–Strogatz

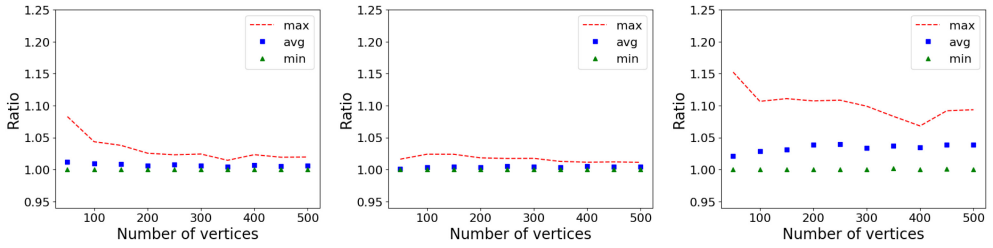
Fig. 7. Performance of BOT, TOP, QoS, CMP, and CMP(Q^*) w.r.t. the terminal selection method using the 2-approximation for ST as a subroutine. Note that the range of the Y-axis is not the same for all plots in the row.



(a) Top-down (b) Composite (c) CMP(Q^*)

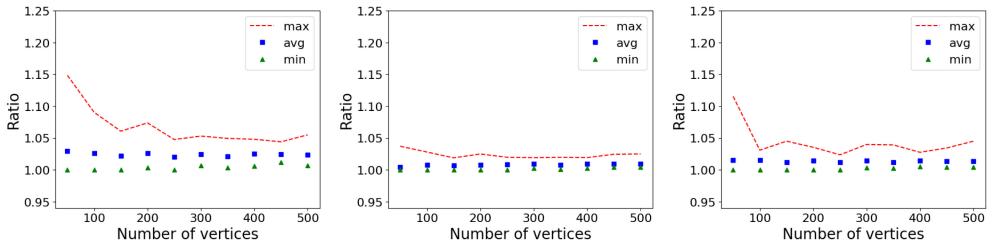
Fig. 8. Performance of TOP, CMP, and CMP(Q^*) on Erdős–Rényi graphs using the 2-approximation for ST as a subroutine.

The most time-consuming part of this experiment was calculating the exact solutions of all MLST instances. It took 88.64 hours to compute all exact solutions. The computation time for network models ER, WS, and BA were 73.8, 7.84, and 7 hours, respectively. Figure 11 shows the time taken to compute the exact solutions (with cost OPT), as a function of the number of levels ℓ. As expected, the running time of the heuristics gets worse as ℓ increases. Note that the y-axes of the graphs in these figures have different scales for different network models. The Erdős–Rényi network model had the highest running time in the worst case.



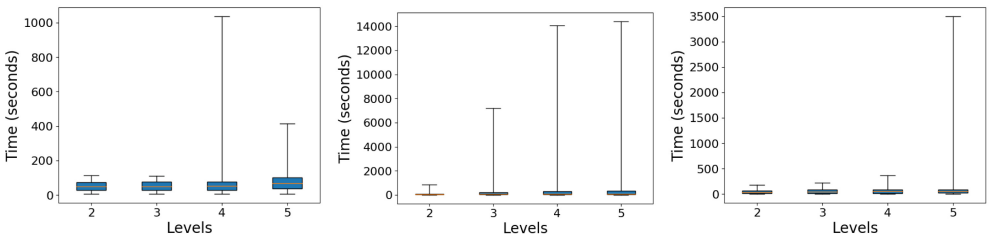
(a) Top-down (b) Composite (c) CMP(Q^*)

Fig. 9. Performance of TOP, CMP, and CMP(Q^*) on Watts–Strogatz graphs.



(a) Top-down (b) Composite (c) CMP(Q^*)

Fig. 10. Performance of TOP, CMP, and CMP(Q^*) on Barabási–Albert graphs using the 2-approximation for ST as a subroutine.



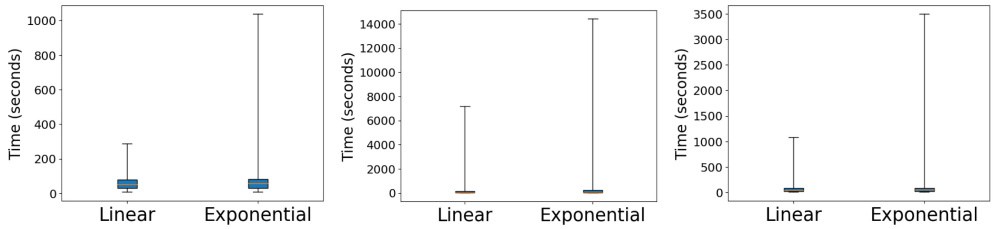
(a) Barabási–Albert (b) Erdős–Rényi (c) Watts–Strogatz

Fig. 11. Experimental running times for computing exact solutions w.r.t. the number ℓ of levels, aggregated over all instances with ℓ levels.

Figure 12 shows the time taken to compute the exact solutions, as a function of the terminal selection method, either LINEAR or EXPONENTIAL. Overall, the running times are slightly worse when the size of the terminal sets decreases exponentially, especially in the worst case.

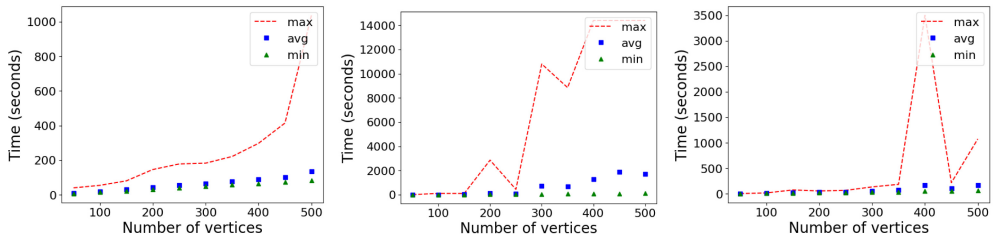
Figure 13 shows the time taken to compute the exact solutions, for each of the network models BA, ER, WS, as a function of the number of vertices $|V|$. Since several instances share the same network size, we show minimum, mean, and maximum values. Note that the y-axes of the graphs in these figures have different scales for different network models. As expected, the running time slightly deteriorated as $|V|$ increased, especially in the worst case.

One motivation for the MLST problem is the visualization of large networks. We have used the algorithms described in this article on a real-world network, originating from a visualization project called UAMap. The underlying data for UAMap is obtained from Google Scholar academic



(a) Barabási–Albert (b) Erdős–Rényi (c) Watts–Strogatz

Fig. 12. Experimental running times for computing exact solutions w.r.t. the terminal selection method, aggregated over all instances with LINEAR or EXPONENTIAL terminal selection.



(a) Barabási–Albert (b) Erdős–Rényi (c) Watts–Strogatz

Fig. 13. Experimental running times for computing exact solutions w.r.t. the graph size $|V|$, aggregated over all instances of $|V|$ vertices.

research profiles and is used to create a weighted research topic graph. The vertices of the graph correspond to self-reported research topics and the edges indicate co-occurring topics in the profiles. The system supports map-based interactive features, including semantic zooming, panning, and searching [5]; see Figure 14. The graph contains 5,947 vertices and 26,695 edges. There are eight levels determined by the popularity of the corresponding research topics. We have run BOT, TOP, and CMP on this graph using the 2-approximation algorithm as a (single-level) ST subroutine. The total cost of BOT, TOP, and CMP is 1702.14, 1701.31, and 1700.66, respectively. Using the solutions of these algorithms, we can visualize the tree of different levels using a similar interactive method mentioned above [5]. Since in every level, we have a tree, we will be able to draw the tree without edge crossings, which is convenient for visualization purposes. Similar to the interactive map visualizations, at a minimum level of zoom, only the most important vertices and edges are visible, and these are provided by the top level tree T_ℓ. Increasing the level of zoom adds in new vertices and edges, and these are provided by the next tree in the filtration until the maximum zoom level is reached and the entire graph is shown.

5 CONCLUSIONS

We presented several heuristics for the MLST problem and analyzed them both theoretically and experimentally. All the software (new heusritcs, approximation algorithms, ILP solvers, experimental data, and analysis) are available online at https://github.com/abureyanahmed/multi_level_steiner_trees.

The heuristics in this article rely on single-level ST subroutines. The composite heuristic CMP achieves the best approximation ratio, as it is the minimum of all possible combinations of single-level ST computations. Importantly, we showed that CMP($\mathcal{Q}^*$) guarantees the same approximation

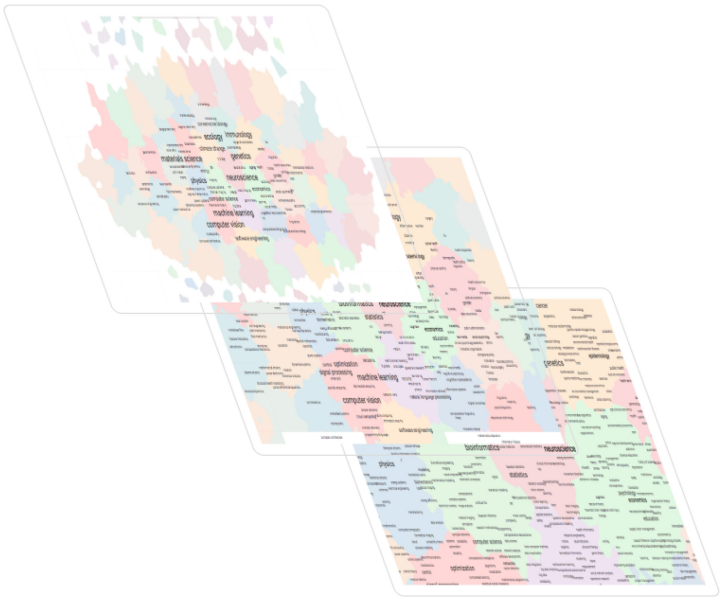


Fig. 14. A multi-level view of UAMap. At the top level, the main topics are shown. More detailed view can be seen in terms of subtopics by zooming in toward the bottom levels.

ratio that CMP can provide, using $O(\ell)$ rather than $O(2^{\ell-1}\ell)$ ST computations. One important question is to consider whether it is possible to directly approximate the MLST problem, without the use of multiple single-level ST subroutines, and whether it is possible to do better than the CMP approximation ratio. Further, it is natural to study whether there are stronger inapproximability results for the MLST problem, compared to the standard ST problem.

Another interesting open problem is whether the approximation ratios t_ℓ (Section 2.2) are tight for any ℓ, and whether the output $\mathbf{y}$ from the LP formulation can help in designing worst-case examples. In particular, even though we have computed the approximation ratio for up to $\ell = 100$ levels, it remains to determine the limit $\lim_{\ell\to\infty} t_\ell$.

As a final remark, even though our investigation focused on the MLST problem, much of the analysis does not depend on the fact that we computed Steiner trees but only that the computed graphs were nested. We thus wonder whether it is possible to generalize our results to other "sparsifiers" (e.g., node-weighted Steiner trees, graph t-spanners).

ACKNOWLEDGMENTS

The authors thank the organizers and participants of the First Heiligkreuztal Workshop on Graph and Network Visualization where work on this problem began.

REFERENCES

[1] Sanjeev Arora. 1998. Polynomial time approximation schemes for Euclidean traveling salesman and other geometric problems. *J. ACM* 45, 5 (1998), 753–782. DOI: https://doi.org/10.1145/290179.290180

[2] Anantaram Balakrishnan, Thomas L. Magnanti, and Prakash Mirchandani. 1994. Modeling and heuristic worst-case performance analysis of the two-level network design problem. *Management Sci.* 40, 7 (1994), 846–867. DOI: https://doi.org/10.1287/mnsc.40.7.846

[3] Albert-László Barabási and Réka Albert. 1999. Emergence of scaling in random networks. *Science* 286, 5439 (1999), 509–512. DOI: https://doi.org/10.1126/science.286.5439.509

[4] Marshall Bern and Paul Plassmann. 1989. The Steiner problem with edge lengths 1 and 2. *Inform. Process. Lett.* 32, 4 (1989), 171–176. DOI: https://doi.org/10.1016/0020-0190(89)90039-2

[5] Randy Burd, Kimberly Andrews Espy, Md Iqbal Hossain, Stephen Kobourov, Nirav Merchant, and Helen Purchase. 2018. GRAM: Global research activity map. In *Proceedings of the International Conference on Advanced Visual Interfaces (AVI'18)*. ACM, New York, NY. DOI: https://doi.org/10.1145/3206505.3206531

[6] Jaroslaw Byrka, Fabrizio Grandoni, Thomas Rothvoß, and Laura Sanità. 2013. Steiner tree approximation via iterative randomized rounding. *J. ACM* 60, 1 (2013), 6:1–6:33. DOI: https://doi.org/10.1145/2432622.2432628

[7] Moses Charikar, Joseph (Seffi) Naor, and Baruch Schieber. 2004. Resource optimization in QoS multicast routing of real-time multimedia. *IEEE/ACM Trans. Netw.* 12, 2 (2004), 340–348. DOI: https://doi.org/10.1109/TNET.2004.826288

[8] Miroslav Chlebík and Janka Chlebíková. 2008. The Steiner tree problem on graphs: Inapproximability results. *Theoret. Comput. Sci.* 406, 3 (2008), 207–214. DOI: https://doi.org/10.1016/j.tcs.2008.06.046

[9] Julia Chuzhoy, Anupam Gupta, Joseph (Seffi) Naor, and Amitabh Sinha. 2008. On the approximability of some network design problems. *ACM Trans. Algorithms* 4, 2 (2008), 23:1–23:17. DOI: https://doi.org/10.1145/1361192.1361200

[10] Paul Erdös and Alfréd Rényi. 1959. On random graphs I. *Publicationes Mathematicae (Debrecen)* 6 (1959), 290–297.

[11] Edgar N. Gilbert and Henry O. Pollak. 1968. Steiner minimal trees. *SIAM J. Appl. Math.* 16, 1 (1968), 1–29. DOI: https://doi.org/10.1137/0116001

[12] Mathias Hauptmann and Marek Karpinski (eds.). 2015. A Compendium on Steiner Tree Problems. Retrieved from http://theory.cs.uni-bonn.de/info5/steinerkompendium/.

[13] Richard M. Karp. 1972. Reducibility among Combinatorial Problems. In *Complexity of Computer Computations*, Raymond E. Miller, James W. Thatcher, and Jean D. Bohlinger (Eds.). Plenum Press, New York, 85–103. DOI: https://doi.org/10.1007/978-1-4684-2001-2_9

[14] Marek Karpinski, Ion I. Mandoiu, Alexander Olshevsky, and Alexander Zelikovsky. 2005. Improved approximation algorithms for the quality of service multicast tree problem. *Algorithmica* 42, 2 (2005), 109–120. DOI: https://doi.org/10.1007/s00453-004-1133-y

[15] Prakash Mirchandani. 1996. The multi-tier tree problem. *INFORMS J. Comput.* 8, 3 (1996), 202–218. DOI: https://doi.org/10.1287/ijoc.8.3.202

[16] Joseph S. B. Mitchell. 1999. Guillotine subdivisions approximate polygonal subdivisions: A simple polynomial-time approximation scheme for geometric TSP, *k*-MST, and related problems. *SIAM J. Comput.* 28, 4 (1999), 1298–1309. DOI: https://doi.org/10.1137/S0097539796309764

[17] Mark E. J. Newman. 2003. The structure and function of complex networks. *SIAM Rev.* 45, 2 (2003), 167–256. DOI: https://doi.org/10.1137/S003614450342480

[18] Tobias Polzin and Siavash Vahdati Daneshmand. 2001. A comparison of Steiner tree relaxations. *Discrete Appl. Math.* 112, 1 (2001), 241–261. DOI: https://doi.org/10.1016/S0166-218X(00)00318-8

[19] Hans Jürgen Prömel and Angelika Steger. 2002. *The Steiner Tree Problem*. Vieweg and Teubner Verlag, Wiesbaden. DOI: https://doi.org/10.1007/978-3-322-80291-0

[20] Gabriel Robins and Alexander Zelikovsky. 2005. Tighter bounds for graph Steiner tree approximation. *SIAM J. Discrete Math.* 19, 1 (2005), 122–134. DOI: https://doi.org/10.1137/S0895480101393155

[21] Duncan J. Watts and Steven H. Strogatz. 1998. Collective dynamics of 'small-world' networks. *Nature* 393, 6684 (1998), 440–442. DOI: https://doi.org/10.1038/30918

[22] Pawel Winter. 1987. Steiner problem in networks: A survey. *Networks* 17, 2 (1987), 129–167. DOI: https://doi.org/10.1002/net.3230170203

[23] Guoliang Xue, Guo-Hui Lin, and Ding-Zhu Du. 2001. Grade of service Steiner minimum trees in the Euclidean plane. *Algorithmica* 31, 4 (2001), 479–500. DOI: https://doi.org/10.1007/s00453-001-0050-6

Received October 2018; revised July 2019; accepted September 2019