Approximation algorithms for the vertex-weighted grade-of-service Steiner tree problem

Faryad Darabi Sahneh
Department of Computer Science, University of Arizona, Tucson, AZ
faryad@cs.arizona.edu

Alon Efrat
Department of Computer Science, University of Arizona, Tucson, AZ
alon@cs.arizona.edu

Stephen Kobourov
Department of Computer Science, University of Arizona, Tucson, AZ
kobourov@cs.arizona.edu

Spencer Krieger
Department of Computer Science, University of Arizona, Tucson, AZ
skrieger@cs.arizona.edu

Richard Spence
Department of Computer Science, University of Arizona, Tucson, AZ
rcspence@cs.arizona.edu

Abstract

Given a graph $G = (V, E)$ and a subset $T \subseteq V$ of terminals, a *Steiner tree* of G is a tree that spans T. In the vertex-weighted Steiner tree (VST) problem, each vertex is assigned a non-negative weight, and the goal is to compute a minimum weight Steiner tree of G. Vertex-weighted problems have applications in network design and routing, where there are different costs for installing or maintaining facilities at different vertices.

We study a natural generalization of the VST problem motivated by multi-level graph construction, the *vertex-weighted grade-of-service Steiner tree problem* (V-GSST), which can be stated as follows: given a graph G and terminals T, where each terminal $v \in T$ requires a facility of a minimum grade of service $R(v) \in \{1, 2, \dots \ell\}$, compute a Steiner tree G' by installing facilities on a subset of vertices, such that any two vertices requiring a certain grade of service are connected by a path in G' with the minimum grade of service or better. Facilities of higher grade are more costly than facilities of lower grade. Multi-level variants such as this one can be useful in network design problems where vertices may require facilities of varying priority.

While similar problems have been studied in the edge-weighted case, they have not been studied as well in the more general vertex-weighted case. We first describe a simple heuristic for the V-GSST problem whose approximation ratio depends on ℓ, the number of grades of service. We then generalize the greedy algorithm of [Klein & Ravi, 1995] to show that the V-GSST problem admits a $(2 \ln |T|)$-approximation, where T is the set of terminals requiring some facility. This result is surprising, as it shows that the (seemingly harder) multi-grade problem can be approximated as well as the VST problem, and that the approximation ratio does not depend on the number of grades of service.

Finally, we show that this problem is a special case of the directed Steiner tree problem and provide an integer linear programming (ILP) formulation for the V-GSST problem.

2012 ACM Subject Classification Theory of computation → Design and analysis of algorithms; Theory of computation → Graph algorithms analysis

Keywords and phrases Approximation algorithms; vertex-weighted Steiner tree; multi-level graph representation; spider decomposition

Digital Object Identifier 10.4230/LIPIcs...

Related Version https://arxiv.org/abs/1811.11700

Acknowledgements We thank Clara Cromey, Anand Vastrad, and Sridhar Mocherla for their help in discussions related to this problem.

1 Introduction

Let $G = (V, E)$ be an undirected, connected graph, and let $T \subseteq V$ be a set of terminals. A *Steiner tree* is a subtree of G that spans T, possibly including other vertices. In the classical *Steiner tree (ST) problem*, each edge of G has a positive weight, and the goal is to find a Steiner tree of minimum weight. The ST problem is NP-hard [19]; it is also APX-hard [3] and cannot be approximated within a factor of 96/95 unless P = NP [9].

In the vertex- (or node-) weighted Steiner tree (VST) problem, the vertices of the graph are assigned positive weights, rather than the edges. The ST problem with edge and/or vertex weights can be formulated as an instance of VST by replacing each edge uv with two edges $\{uw, wv\}$, where the weight of w equals the weight of uv. Vertex-weighted problems have many applications in network routing, where there may be different costs for installing or maintaining facilities at different vertices. The VST problem is provably harder than ST, and cannot be approximated within a factor of $(1-\varepsilon)\ln|T|$ unless P = NP [14], via a simple reduction from the set cover problem [21]. There are nearly best-possible approximation algorithms for the VST problem that achieve an $O(\log|T|)$ approximation ratio [17, 21].

In many applications, the terminals may require different levels, priorities, or grades of service [2, 7, 10, 23, 24]. For example, when connecting cities with a network, larger cities (hubs) often require higher-quality facilities than smaller ones.

1.1 Problem definition

We state the problem naturally in terms of facilities and grades of service. Given an undirected graph $G = (V, E)$, let $T \subseteq V$ be a subset of terminals. Each terminal $v \in T$ has a *required* grade of service $R(v) \in \{1, 2, \ldots, \ell\}$. The goal is to install facilities on a subset of vertices, where each terminal $v \in T$ contains a facility of grade $R(v)$ or higher, while ensuring connectivity between vertices requiring the same grade of service or higher. Given $v \in V$ and $1 \le i \le \ell$, let $c_i(v)$ denote the cost of installing a facility of grade i on vertex v. Naturally, facilities of a higher grade of service are more costly, so we stipulate $0 \le c_1(v) \le c_2(v) \le \ldots \le c_\ell(v)$ for all $v \in V$.

▶ **Definition 1** (Vertex-weighted Grade-of-Service Steiner Tree (V-GSST) problem). *Given an undirected graph $G = (V, E)$, a set of terminals $T \subseteq V$ with required grades of service $R : T \to \{1, 2, \ldots, \ell\}$, and installation costs $c_i(v)$, compute a Steiner tree G' of G with (integer) assigned grades of service $y(v)$ such that the following hold:*

- *For all $v \in T$, the assigned grade of service of v is greater than or equal to its required grade of service $R(v)$, i.e., $1 \le R(v) \le y(v) \le \ell$ for all $v \in T$.*
- *For all terminals $u, v \in T$, the u-v path in G' uses vertices with assigned grade of service* $\min(R(u), R(v))$ *or higher. That is, for each w along the u-v path in G', we have $y(w) \ge \min(R(u), R(v))$.*

The cost of a solution is defined as the sum of the costs of all installed facilities in G', namely $\sum_{v \in V(G')} c_{y(v)}(v)$. The V-GSST problem *is to find a minimum cost subtree G^* of cost* OPT.

We assume w.l.o.g. that edges have zero cost, as an instance with edge and vertex costs can be converted to an instance with only vertex costs. We may define $c_0(v) = 0$ and $y(v) = 0$ for vertices $v \in V \setminus V(G')$; that is, no facility is installed on v. Additionally, because edges have zero cost, a solution can be found given only $y(v)|_{v \in V}$ by finding a spanning tree over the vertices $\{v \mid y(v) = \ell\}$, then iteratively contracting the tree and computing a spanning tree over vertices with $y(v) = \ell - 1$, and so on. The case $\ell = 1$ is the VST problem.

In the VST problem, it is conventional to assume that terminals have zero cost, as the terminals must be involved in any feasible solution [17]. Similarly for the V-GSST problem, we can assume w.l.o.g. that $c_1(v) = \ldots = c_{R(v)}(v) = 0$ for each $v \in T$; this ignores the "required" cost $\sum_{v \in T} c_{R(v)}(v)$, which can be helpful in assessing the quality of a solution. In both problems, an instance with positive terminal costs can be converted to an instance with zero terminal costs. For the V-GSST problem, one can create a dummy vertex v' for each $v \in T$ with zero installation cost, add edge vv', and use v' as a terminal instead of v. Lastly, we assume w.l.o.g. that there exists $v \in T$ with $R(v) = \ell$; otherwise ℓ can be reduced.

1.2 Related work

The (edge-weighted) ST problem admits a simple 2-approximation [15], via a minimum spanning tree of the metric closure[1] of T. The linear program (LP)-based approximation algorithm of Byrka et al. [4] gives a ratio of $\rho = \ln 4 + \varepsilon < 1.39$. Details about the ST problem and its variants can be found in [18, 29, 32], with more edge-weighted network design problems in [16].

Klein and Ravi [21] give a greedy $2 \ln |T|$-approximation to the VST problem (see Section 3) over terminals T. Guha and Khuller [17] improve the approximation ratio to $1.5 \ln |T|$ via minimum-weight "3+ branch spiders;" however, the algorithm is not practical for large graphs. Demaine et al. [12] showed that the VST problem admits a polynomial time constant approximation when restricted to planar graphs. Other variants of the VST problem have also been studied, including bi-criteria [25], multi-commodity [22], k-connectivity [27], and degree constrained [30]. Exact or near-exact approaches for VST based on Lagrangian relaxation have also been studied [11, 13].

Chekuri et al. [8] study the similar node-weighted buy-at-bulk network design problem, defined in terms of sending flows δ_i to node pairs s_i-t_i. The cost of routing x_v flow through a node v is given by a sub-additive function $f_v()$. The authors show that the single-source problem (NSS-BB) with non-uniform flow costs admits an $O(\log |T|)$-approximation by giving a randomized algorithm for NSS-BB, then derandomizing it using an LP relaxation. Another somewhat related problem is the *online* vertex-weighted Steiner tree problem, in which the terminals T arrive online. At any stage, a subgraph must connect all terminals that have arrived thus far. Naor et al. [26] describe a randomized $O(\log |V| \log^2 |T|)$-approximation algorithm to the online problem.

Several results are known on multi-level or grade-of-service Steiner tree problems, where edges are weighted. Balakrishnan et al. [2] give a $(4/3)\rho$-approximation algorithm for the 2-level network design problem with proportional edge costs. Charikar et al. [7] describe a simple 4ρ-approximation for the Quality-of-Service (QoS) Multicast Tree problem with proportional edge costs (termed the *rate model*), which is improved to $e\rho$ through randomized doubling. Karpinski et al. [20] use an iterative contraction scheme to obtain a 2.454ρ-approximation. Ahmed et al. [1] further improve the approximation ratio to 2.351ρ. Xue et al. [33] show that the grade-of-service Steiner tree problem in the Euclidean plane admits $\frac{4}{3}\rho$ and $\frac{5+4\sqrt{2}}{7}\rho$-approximations for 2 and 3 grades, respectively.

Node-weighted problems on graphs can often be converted to equivalent edge-weighted problems, by requiring that the converted graph is directed. Segev [31] gives a simple reduction from VST to the directed Steiner tree (DST) problem, where each edge $uv \in E$ is

[1] Given $G = (V, E)$, the *metric closure* of $T \subseteq V$ is the complete graph $K_{|T|}$, where edge weights are equal to the lengths of corresponding shortest paths in G.

replaced with two directed edges (u, v) and (v, u), and the weight of edge (u, v) equals the weight of its incoming vertex v. The DST problem with k terminals admits a $i(i-1)k^{1/i}$-approximation in time $O(n^i k^{2i})$ for fixed $i \geq 1$ [6] using a recursive greedy approach, which implies a polynomial time $O(k^\varepsilon)$-approximation for fixed $\varepsilon > 0$. By setting $i = \log k$, DST can be $O(\log^2 k)$-approximated in quasi-polynomial time. We show that the V-GSST problem is also a special case of DST; see Appendix A.

1.3 Our contributions

We consider simple top-down and bottom-up approaches (Section 2) and show that the top-down approach is an ℓ-approximation to the V-GSST problem, where ℓ is the number of grades of service, if one is allowed access to a VST oracle. If one replaces an oracle with an approximation, this gives a polynomial time $O(\ell \log |T|)$-approximation to the V-GSST problem. However, the bottom-up approach can perform arbitrarily badly.

The main result is the following:

▶ **Theorem 2.** *There is a polynomial time $(2 \ln |T|)$-approximation algorithm for the V-GSST problem with arbitrary costs $c_i(v)$ for each vertex and grade of service.*

The algorithm, which we refer to as GreedyVGSST (Section 3), relies on a generalization of the methods by Klein and Ravi [21]. This result is surprising, as it shows that the (seemingly harder) multi-level problem can be approximated as well as the VST problem, and the approximation ratio does not depend on the number of grades of service. Unless P = NP, the approximation ratio is within a constant factor of the best possible.

The GreedyVGSST algorithm maintains a set of "grade-respecting trees" and carefully merges a subset of these trees so that the newly-formed tree is also grade respecting. The main tool of the analysis is based on the existence of a "rooted spider decomposition." Similar graph decompositions are often used in network design problems [5, 8, 17, 21, 22, 27, 28, 30], and we expect them to be applicable to other multi-level network design problems.

Finally, we show that the V-GSST problem is a special case of the directed Steiner tree problem (Appendix A), and provide an integer linear programming (ILP) formulation (Appendix C).

2 Top-down and bottom-up approaches for V-GSST

In this section, we describe two simple heuristics, the *top-down* and *bottom-up* approaches.

2.1 Top-down approach

Assume an oracle can compute a minimum-weight VST for an input graph G over terminals T with vertex costs $c(\cdot)$, denoted $E' = VST(G, c, T)$. A top-down approach is as follows: compute a VST of G over terminals $v \in T$ with the highest required grade of service $(R(v) = \ell)$, using vertex costs $c_\ell(v)$, and install a facility of grade ℓ on each vertex spanned by this VST. The cost incurred at this step, which we denote by TOP_ℓ, equals the total cost of installing these facilities of grade ℓ. Then, contract this tree into a single terminal with required grade $\ell - 1$ and zero cost. Compute a VST of G over the remaining terminals satisfying $R(v) = \ell - 1$, and install facilities of grade $\ell - 1$ on each vertex spanned by this VST. Continue this process iteratively until a feasible solution is obtained. This approach is summarized in Algorithm 1.

Algorithm 1 Top-down approach

```
procedure TopDownVGSST(G, c, R)
    for i = ℓ, ..., 1 do
        E_i := VST(G, c_i, {v : R(v) ≥ i})
        Set y(v) := i for each v spanned by E_i
        if i > 1 then
            Contract subtree E_i to a single terminal t with R(t) = i − 1 and zero cost
        end if
    end for
end procedure
```

Let TOP be the cost of the V-GSST returned by the top-down approach (Algorithm 1), let TOP_i be the total cost of all vertices in the returned solution containing a facility of grade equal to i, so that $\text{TOP} = \sum_{i=1}^{\ell} \text{TOP}_i$.

▶ **Proposition 3.** *The top-down heuristic (with an oracle) returns a solution whose cost is not worse than* $\ell \cdot \text{OPT}$.

The proof is given in Appendix B. The ratio of ℓ is tight as shown by example in Figure 1. In this example, the top-down approach avoids a non-terminal "hub" v whose cost is only slightly more than the cost of all other non-terminals. If an oracle for VST is replaced with an $O(\log |T|)$-approximation, then this gives a polynomial time $O(\ell \log |T|)$-approximation to the V-GSST problem.

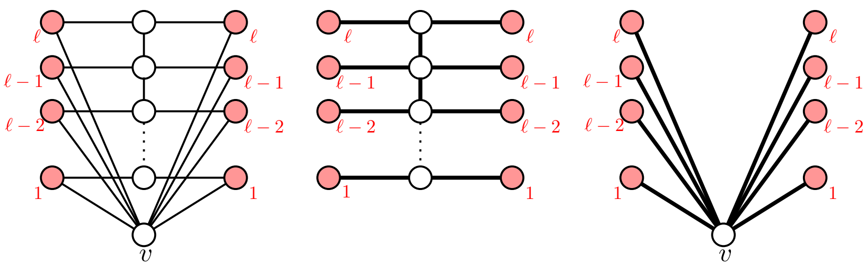


Figure 1 The ℓ-approximation for the top-down V-GSST heuristic is asymptotically tight. *Left:* Input graph, with terminals and required grades of service in red. The cost of installing a facility of any grade of service on v is $1+\varepsilon$ (i.e., $c_1(v) = c_2(v) = \ldots = c_\ell(v) = 1+\varepsilon$), while the cost of installing a facility of any grade of service on any other non-terminal is 1. *Center:* The top-down approach installs a facility of cost 1 on each of the ℓ non-terminals except v, giving $\text{TOP} = \ell$. *Right:* The minimum cost solution installs a facility of grade ℓ on v, giving $\text{OPT} = 1+\varepsilon$. Hence $\frac{\text{TOP}}{\text{OPT}} = \frac{\ell}{1+\varepsilon} \approx \ell$.

2.2 Bottom-up approach

An analogous "bottom-up" approach is to compute a VST over the set of all terminals T, using cost function $c_\ell(\cdot)$. Installing facilities of grade of service equal to ℓ on each vertex of this VST produces a valid solution. We can then demote these vertices' assigned grade of service to locally improve the cost of the solution. However, this poses challenges as $c_\ell(\cdot)$ may add vertices v of grade ℓ which cannot be demoted, where $c_\ell(v) \gg c_{\ell-1}(v)$.

3 The GreedyVGSST algorithm

We first review the $(2 \ln |T|)$-approximation algorithm by Klein and Ravi [21] for the VST problem (referred to as KR) and then describe the generalization to the V-GSST problem. The KR algorithm maintains a forest $\mathcal{F}$; initially, each terminal is a singleton tree. At each iteration, a vertex v as well as a subset $\mathcal{S} \subseteq \mathcal{F}$ consisting of two or more trees ($|\mathcal{S}| \geq 2$) is merged to form a single tree, with the objective of minimizing the *quotient cost* $\frac{c(v)+\sum_{\mathcal{T}\in\mathcal{S}} d(v,\mathcal{T})}{|\mathcal{S}|}$. Here, $d(v, \mathcal{T})$ is the shortest distance from v to any vertex in the tree $\mathcal{T}$, excluding endpoint costs. For any given vertex v, an optimal subset $\mathcal{S}$ and its corresponding quotient cost can be found in polynomial time, as the only subsets $\mathcal{S}$ that need to be considered are those consisting of the 2, 3, ..., $|\mathcal{F}|$ nearest trees from v. The algorithm terminates when $|\mathcal{F}| = 1$.

3.1 Setup

We use the observation that given an instance of V-GSST consisting of a graph G, required grades of service $R(v)$, and vertex costs $c_i(v)$, there exists an optimal solution G^* (in terms of assigned grades of service $y^*(\cdot)$) such that from any vertex $r \in T$ with $R(r) = \ell$, the path from r to any other terminal uses vertices of non-increasing assigned grades of service.

▶ **Definition 4** (Grade-Respecting Tree (GRT)). *Let G be a graph, and $\mathcal{T}$ be a subtree of G. Let $y : V(G) \to \{0, 1, \ldots, \ell\}$ be a labeling function that assigns grades of service to vertices in G. Let $r \in \mathcal{T}$ be a root vertex. We say that $\mathcal{T}$ is a* grade-respecting tree *rooted at r if, for all $v \in V(\mathcal{T})$, the path from r to v in $\mathcal{T}$ uses vertices of non-increasing grade of service.*

Our generalization to the V-GSST problem relies on maintaining a set $\mathcal{F}$ of GRTs. The notion is similar to that of the rate-of-service Steiner tree [33] or QoS Multicast Tree [7] except that we grow and maintain a collection of such trees during the algorithm. The trees in $\mathcal{F}$ are not necessarily vertex-disjoint as the same vertex may appear in different GRTs; hence, we avoid the term "forest."

For $v \in V(G)$, let $y(v)$ denote the grade of service of v at the current iteration of the algorithm. Initially, $y(v) = R(v)$ for each $v \in T$, and $y(v) = 0$ for $v \in V(G) \setminus T$. The grades $y(\cdot)$ are updated in each iteration, and are returned as output. On each iteration, a subset of the current set $\mathcal{F}$ of GRTs is greedily chosen and connected via their roots to form a new GRT $\mathcal{T}_{new}$. The set $\mathcal{F}$ is updated, as well as the root and grade of service assignments $y(\cdot)$ for vertices in $\mathcal{T}_{new}$. The size of $|\mathcal{F}|$ is strictly decreasing at each iteration; once $|\mathcal{F}| = 1$, the resulting GRT is returned as a feasible solution to the V-GSST problem.

To decide which trees to connect, define ℓ cost functions $w_1, w_2, \ldots, w_\ell : V(G) \to \mathbb{R}_{\geq 0}$ with the interpretation that $w_i(v)$ denotes the "incremental" cost of upgrading vertex v from its current grade of service $y(v)$ to i. Initially, $w_i(v) = c_i(v)$ for all $i \in \{1, \ldots, \ell\}$ and $v \in V$. The cost $w_i(v)$ is reduced when v is included in $\mathcal{T}_{new}$ to reflect that we have already paid a certain cost for v. Additionally, let $d_i(u, v)$ be the cost of a shortest path from u to v using cost function $w_i(\cdot)$, not including the costs of the endpoints u and v. As the costs $w_i(\cdot)$ are updated at each iteration, the distances $d_i(u, v)$ also update at each iteration.

3.2 Initialization

Initialize $\mathcal{F}$ so that each terminal $v \in T$ is its own GRT; initially $|\mathcal{F}| = |T|$. Initialize $y(v) = R(v)$ for all $v \in T$, and $y(v) = 0$ for all $v \in V \setminus T$. Lastly, initialize $w_i(v) = c_i(v)$ for all $i \in \{1, \ldots, \ell\}$ and $v \in V$.

3.3 Iteration step

Each iteration consists of selecting a *root GRT* $\mathcal{T} \in \mathcal{F}$ rooted at r, a *center* $v_c \in V$ (the center may or may not be r), an integer $i \leq R(r)$ representing the grade of service that v_c is "promoted" to, and a nonempty subset $\mathcal{S} = \{\mathcal{T}_k\} \subset \mathcal{F}$ of GRTs, such that $R(r_k) \leq i$ for all roots r_k associated with $\mathcal{S}$. By properly connecting r to v_c using facilities of grade i, then connecting v_c to each root r_k using facilities of grade $R(r_k)$, we can substitute the $|\mathcal{S}| + 1$ GRTs with a new GRT $\mathcal{T}_{new}$ rooted at r. Figure 2 illustrates an iteration step.

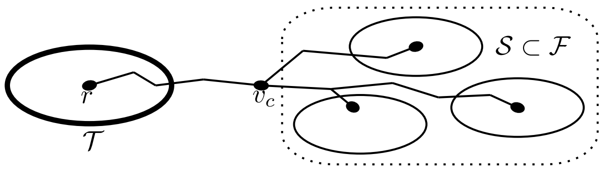


Figure 2 Illustration of an iteration step in GreedyVMLST for some set $\mathcal{S}$ with $|\mathcal{S}| = 3$. The root of the newly formed tree $\mathcal{T}_{new}$ is r.

At each iteration, we select a root GRT, v_c, i, and $\mathcal{S}$ in order to minimize the *cost-to-connectivity* ratio γ, defined as follows:

$$\gamma = \frac{d_i(r, v_c) + w_i(v_c) + \sum_{k=1}^{|\mathcal{S}|} d_{R(r_k)}(v_c, r_k)}{1 + |\mathcal{S}|} \tag{1}$$

The expression $d_i(r, v_c) + w_i(v_c)$ is the cost of upgrading all vertices (including v_c) along a shortest r-v_c path to grade i. The summation $\sum_{k=1}^{|\mathcal{S}|} d_{R(r_k)}(v_c, r_k)$ represents the incremental cost of upgrading the vertices from v_c to each root r_k to the appropriate grade of service; this may overcount costs of vertices that appear on multiple center-to-root paths. The denominator $1 + |\mathcal{S}|$ represents the number of GRTs that are merged.

Once we have selected r, v_c, i, and $\mathcal{S}$ that minimizes γ, we update the assigned grades $y(\cdot)$ for vertices spanned by $\mathcal{T}_{new}$ as follows: for each v on a shortest r-v_c path, set $y(v) \leftarrow \max(y(v), i)$. For each v on a shortest v_c-r_k path, set $y(v) \leftarrow \max(y(v), R(r_k))$. If v appears on multiple center-to-root paths, then $y(v)$ is set to the maximum over all such $R(r_k)$ grades that v connects.

The GreedyVGSST algorithm is summarized in Algorithm 2. Figure 3 shows a concrete example of the execution of GreedyVGSST, using proportional vertex costs to simplify the presentation. Recall we assume $c_1(v) = \ldots = c_{R(v)}(v) = 0$ for each $v \in T$.

Algorithm 2 The GreedyVGSST algorithm

procedure GreedyVGSST(G, c, R)
  Initialize $\mathcal{F}$ so that each terminal $v \in T$ is a singleton GRT with $y(v) = R(v)$
  For each $i = 1, \ldots, \ell$ and $v \in V$, initialize $w_i(v) = c_i(v)$
  while $|\mathcal{F}| > 1$ **do**
    Find root GRT $\mathcal{T} \in \mathcal{F}$, center v_c, integer i, and $\mathcal{S} \subset \mathcal{F}$ minimizing γ.
    Update $y(v)$ for v on the r-v_c path, as well as for center-to-root paths
    Delete $\mathcal{T}$ and all GRTs in $\mathcal{S}$ from $\mathcal{F}$. Add GRT $\mathcal{T}_{new}$ to $\mathcal{F}$.
    Update weight functions: for $v \in V(\mathcal{T}_{new})$, set $w_j(v) = 0$ if $j \leq y(v)$, and $w_j(v) = w_j(v) - w_{y(v)}(v)$ if $j > y(v)$.
  end while
end procedure

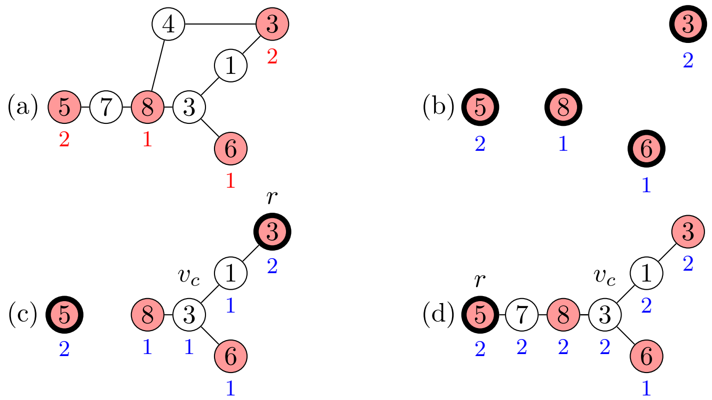


Figure 3 Illustration of GreedyVGSST. (a): Input graph with $\ell = 2$, with terminals and positive required grades of service $R(\cdot)$ shown in red, and proportional costs (e.g. the terminal v with cost 8 has $(c_1(v), c_2(v)) = (0, 8)$, while the non-terminal u with cost 3 has $(c_1(u), c_2(u)) = (3, 6)$). (b): Initial set of singleton GRTs $\mathcal{F}$ with $|\mathcal{F}| = 4$. (c): Result after choosing r and v_c as shown, $i = 1$, and $\mathcal{S}$ of size 2, which minimizes γ. The value of γ is $\gamma = \frac{1\cdot1+1\cdot3+0+0}{1+2} = \frac{4}{3}$. Note that $w_1(v_c), w_2(v_c) = (3, 6)$ initially; after the first iteration, we have $(w_1(v_c), w_2(v_c)) = (0, 3)$. (d): Choosing r and v_c as shown, $i = 2$, and $\mathcal{S}$ of size 1 minimizes γ. In this case, $d_2(r, v_c) = 2 \cdot 7 + 1 \cdot 8 = 22$, $w_2(v_c) = 3$, and $d_2(v_c, r_1) = 1$ where r_1 is the root of the single tree in $|\mathcal{S}|$. The value of γ is $\gamma = \frac{22+3+1}{1+1} = 13$. Since $|\mathcal{F}| = 1$, GreedyVGSST terminates after two iterations with a cost of $2 \cdot 7 + 1 \cdot 8 + 2 \cdot 3 + 2 \cdot 1 = 30$.

Observe that, as edges are not weighted, the actual trees in $\mathcal{F}$ at each iteration are not very important until after the final iteration; it suffices to only keep track of roots in $\mathcal{F}$ along with the vertices associated with each root.

▶ **Lemma 5.** *A choice of r, v_c, i, and $\mathcal{S}$ that minimizes γ can be found in polynomial time.*

Proof. For a fixed center v_c and integer i, sort all trees in $\mathcal{F}$ whose root r_k has $R(r_k) \le i$ by their "distance" to v_c, namely $d_{R(r_k)}(v_c, r_k)$. The best choice for subset $\mathcal{S}$ can be found through checking only subsets with the nearest $2, 3, \ldots, |\mathcal{F}|$ trees. Therefore, for fixed v_c and i, we can find $\mathcal{S}$ that minimizes $\frac{w_i(v_c)+\sum_{k=1}^{|\mathcal{S}|} d_{R(r_k)}(v_c,r_k)}{|\mathcal{S}|}$ in polynomial time. If $i > \max_k R(r_k)$, then we can improve γ by setting $i = \max_k R(r_k)$.

Lastly, find a GRT $\mathcal{T}' \in \mathcal{F}$ with root r' satisfying $R(r') > i$, whose root is closest to v_c under cost function $w_i(\cdot)$. If such a GRT exists, and if choosing r' as the root lowers γ, then accept this GRT as the root GRT. Otherwise, use one of the GRTs in $\mathcal{S}$ whose root has grade equal to i as the root tree, remove it from $\mathcal{S}$, set it as the root GRT $\mathcal{T}$, and set i as the maximum grade of the remaining GRT roots in $\mathcal{S}$. Therefore, for a fixed center vertex v_c and grade i, we can find a root r and subset $\mathcal{S}$ that minimizes γ. As there are $|V|\ell$ choices for v_c and i, a choice of r, v_c, i, and $\mathcal{S}$ that minimizes γ can be found in polynomial time. ◀

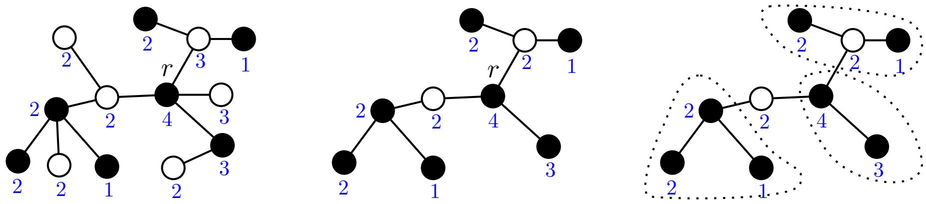
Compared to the KR algorithm, an iteration in the GreedyVGSST algorithm requires determining two new elements: a root GRT $\mathcal{T}$, and an integer i representing the grade of service that v_c is upgraded to. However, the above proof indicates that a GreedyVGSST iteration is only ℓ times more expensive than one for the KR algorithm.

3.4 Analysis of GreedyVGSST

The analysis of the KR algorithm [21] uses a *spider decomposition* of the optimal VST solution. First, we introduce the notion of a "locally optimal solution" with respect to a subset M of vertices. Recall that $y(v)$ denotes the assigned grade of service of v.

▶ **Definition 6** ($M-$Optimized GRT). *Let $\mathcal{T}$ be a GRT rooted at r, and let $M \subseteq V(\mathcal{T})$ such that $r \in M$. Then $\mathcal{T}$ is* M-optimized *if all leaves of $\mathcal{T}$ are contained in M, and for any vertex $v \in V(\mathcal{T}) \setminus M$, we have that $y(v) = \max y(w)$ for all $w \in M$ in the subtree rooted at v.*

Thus, an M-optimized GRT does not unnecessarily use higher grades of service to reach vertices in M from r; see Figure 4a-4b for an example. There necessarily exists a minimum V-GSST G^* that is T-optimized due to the following simple argument: if there is a vertex $v \notin T$, for which $y^*(v) > y^*(w)$ for all $w \in T$ in the subtree rooted at v, then demoting $y^*(v)$ to equal $\max_w y^*(w)$ over all w in the subtree rooted at v leads to a solution whose cost is less than or equal to that of G^*. Further, given a GRT $\mathcal{T}$ rooted at r, and a set $M \subseteq V(\mathcal{T})$ with $r \in M$, it is not difficult to compute an M-optimized subtree of $\mathcal{T}$.



(a) A GRT $\mathcal{T}$ with vertices in M shown in black. **(b)** An M-optimized GRT, obtained from Fig. 4a. **(c)** Rooted spider decomposition of the GRT from Fig. 4b.

Figure 4 A GRT with $|M| = 7$ and assigned grades $y(\cdot)$, an M-optimized GRT, and a rooted spider decomposition.

The following definition is due to Klein and Ravi [21]:

▶ **Definition 7** (Spider [21]). *A* spider *is a tree where at most one vertex has degree greater than two. A spider is identified by its* center*, a vertex from which all paths to the leaves of the spider are vertex-disjoint. A* nontrivial spider *is a spider with at least two leaves.*

A *foot* of a spider is a leaf; if the spider has at least three leaves, then its center is unique, and is also a foot. Klein and Ravi [21] show that given a connected graph $G = (V, E)$ and a subset $M \subseteq V$ of vertices, a set of vertex-disjoint nontrivial spiders can be found such that the union of the feet of the spiders contains M. We generalize the notion of a spider to the multi-grade setting, which we refer to as a *rooted spider*.

▶ **Definition 8** (Rooted spider). *A* rooted spider *is a GRT $\mathcal{T}$ which is also a spider. It is identified by a* center *v_c and a* root *r, such that the following properties hold:*

- *The root r is either the center or a leaf of $\mathcal{T}$, and the path from r to every vertex in the spider uses vertices of non-increasing grade of service $y(\cdot)$*
- *The paths from v_c to each non-root leaf of $\mathcal{T}$ are vertex-disjoint and use non-increasing grades of service*

The resulting tree in Figure 3 is a rooted spider whose root is distinct from its center.

▶ **Lemma 9.** *Let $\mathcal{T}$ be an M-optimized GRT rooted at r, where $|M| \geq 2$ and $r \in M$. Then $\mathcal{T}$ can be decomposed into vertex-disjoint rooted spiders such that the rooted spider leaves and roots belong to M (a rooted spider center may or may not belong to M), and the rooted spider leaves, roots, and centers cover the set M.*

Proof. We use induction on $|M|$. For the base case $|M| = 2$, the decomposition consists of a single rooted spider, namely the path in $\mathcal{T}$ from r to the other vertex in M, where r is also the root of its rooted spider.

For $|M| \geq 3$, find a vertex $v \in \mathcal{T}$ with the property that the subtree $\mathcal{T}'$ rooted at v contains at least two vertices in M, such that v is furthest from r by number of edges. If $v = r$, then $\mathcal{T}$ is already a rooted spider, as no other vertex in $V(\mathcal{T}) \setminus \{r\}$ can have degree greater than 2; if there existed $w \neq r$ with degree greater than 2, then the subtree rooted at w has at least two leaves (which belong to M), contradicting the choice of $v = r$. In this case, set r to be the root and center of its rooted spider.

Thus, we assume $v \neq r$. The vertex v and its subtree forms a rooted spider with center v. If $v \in M$, then set v to be the root of its rooted spider. If $v \notin M$, then since $\mathcal{T}$ is M-optimized, there exists $w \in M$ in the subtree rooted at v with grade of service $y(w) = y(v)$; set w to be the root of its rooted spider. Remove this rooted spider from the original tree $\mathcal{T}$, as well as the edge from v to its parent, to produce a smaller tree $\mathcal{T}'$.

Let $M' \subset M$ be the set of vertices in M that remain in $\mathcal{T}'$ upon removing the subtree rooted at v. If $|M'| = 0$, then the subtree rooted at v is the only rooted spider, giving a valid decomposition. If $|M'| = 1$, then $M' = \{r\}$, so connecting r to v produces a single rooted spider with root r and center v as $y(r) \geq y(v)$. Otherwise $|M'| \geq 2$, so we may prune the r-v path so that $\mathcal{T}'$ is an M'-optimized GRT. By the induction hypothesis, $\mathcal{T}'$ can be decomposed into vertex-disjoint rooted spiders over M'. ◀

Figure 4c gives a rooted spider decomposition of the M-optimized GRT from Figure 4b.

▶ **Corollary 10.** *Let $\mathcal{T}$ be an M-optimized GRT. Consider a rooted spider decomposition containing s rooted spiders $\mathcal{X}_1, \ldots, \mathcal{X}_s$, generated using the method in the proof of Lemma 9. Let r_j be the root of the $\mathcal{X}_j$, and let $L_j = (M \cap V(\mathcal{X}_j)) \setminus r_j$ be the vertices in M contained in $\mathcal{X}_j$, not including the root r_j. Then $\sum_{j=1}^{s}(1 + |L_j|) = |M|$.*

Proof. This statement follows as every vertex in $v \in M$ is either a root of its rooted spider, or is reachable from its spider's center. In particular, the path from the rooted spider's center v to any leaf $w \in M$ does not encounter any other vertices in M, as this would contradict the choice of v when computing such a decomposition. ◀

For the next lemma, we define the following notation. Let $\mathcal{F}_n$ $(n \geq 1)$ denote the set of GRTs at the beginning of iteration n, and let M_n denote the set of GRT roots at the beginning of iteration n. Thus, $|\mathcal{F}_n| = |M_n|$, and $|\mathcal{F}_1| = |T|$. Let G^* be a minimum V-GSST with assigned grades $y^* : V \to \{0, 1, \ldots, \ell\}$ and cost OPT.

Consider the M_n-optimized GRT obtained by optimizing G^* with respect to M_n and y^*. By Lemma 9, this GRT contains a rooted spider decomposition containing s rooted spiders $\mathcal{X}_1, \ldots, \mathcal{X}_s$. Consider the j^{th} rooted spider $\mathcal{X}_j$ $(1 \leq j \leq s)$ with root r_j, center $v_{c,j}$, and terminals $L_j \subset M_n$ not including its root r_j. Let $c(\mathcal{X}_j)$ be the cost of the vertices in $\mathcal{X}_j$ within G^*, given by $c(\mathcal{X}_j) = \sum_{v \in V(\mathcal{X}_j)} c_{y^*(v)}(v)$. On the current iteration, a candidate for GreedyVGSST is to select root $r = r_j$, center $v_c = v_{c,j}$, $i = y^*(v_c)$, and $\mathcal{S}$ the set of GRTs in $\mathcal{F}_n$ whose root is in L_j (so that $|\mathcal{S}| = |L_j|$). Let $\widehat{c_j}$ be the cost that GreedyVGSST computes for this candidate (i.e., the numerator in the expression for γ, eq. (1)). We show in Lemma 11 that this computed cost is not more than the cost of the j^{th} rooted spider $\mathcal{X}_j$.

▶ **Lemma 11.** *Consider the j^{th} rooted spider $\mathcal{X}_j$ in a rooted spider decomposition of the M_n-optimized GRT of G^*, and consider the candidate choice r, v_c, i, and $\mathcal{S}$ as described above with computed cost $\widehat{c_j}$. Then $\widehat{c_j} \leq c(\mathcal{X}_j)$.*

Proof. Let $p(u,v)$ denote the u-v path in the rooted spider $\mathcal{X}_j$ not including endpoints u and v, and let $c^*(u,v) = \sum_{v \in p(u,v)} c_{y^*(v)}(v)$ denote the sum of vertex costs along path $p(u,v)$. Because the paths from $v_{c,j}$ to the root r_j or to each $w \in L_j$ are vertex-disjoint by Def. 8, we have

$$c(\mathcal{X}_j) \geq c^*(r_j, v_{c,j}) + \sum_{k=1}^{|L_j|} c^*(v_{c,j}, r_k).$$

However, $\widehat{c_j}$ considers the minimum-cost vertex-weighted paths between r_j and $v_{c,j}$, as well as from $v_{c,j}$ to each $r_k \in L_j$. Thus $c(\mathcal{X}_j) \geq \widehat{c_j}$ as desired. ◀

Let $h_n \geq 2$ denote the number of GRTs in $\mathcal{F}_n$ that are merged on the n^{th} iteration, including the root GRT. Let ΔC_n denote the actual cost incurred on the n^{th} iteration of GreedyVGSST; for example, $\Delta C_1 = 4$ and $\Delta C_2 = 26$ in the example in Fig. 3. Let γ_n denote the minimum cost-to-connectivity ratio *computed by the* GreedyVGSST *algorithm* on the n^{th} iteration (e.g. $\gamma_1 = \frac{4}{3}$ and $\gamma_2 = 13$ in Fig. 3).

▶ **Lemma 12.** *For each iteration $n \geq 1$ of* GreedyVGSST, *we have* $\dfrac{\Delta C_n}{h_n} \leq \dfrac{\text{OPT}}{|\mathcal{F}_n|}$.

Proof. Fix an iteration $n \geq 1$, and consider the M_n-optimized GRT obtained from G^*. Recall that for $n = 1$, G^* is already T-optimized ($M_1 = T$). By Theorem 9, there exists a rooted spider decomposition over M_n, containing $s \geq 1$ rooted spiders $\mathcal{X}_1, \ldots, \mathcal{X}_s$.

As GreedyVGSST aims to minimize γ, we necessarily have

$$\gamma_n \leq \frac{\widehat{c_j}}{1 + |L_j|} \underbrace{\leq}_{\text{Lemma 11}} \frac{c(\mathcal{X}_j)}{1 + |L_j|}. \tag{2}$$

The computed cost in γ_n is greater than or equal to ΔC_n, as the computed cost may overcount vertex costs appearing on multiple center-to-root paths. Hence (2) implies $\frac{\Delta C_n}{h_n} \leq \gamma_n \leq \frac{c(\mathcal{X}_j)}{1+|L_j|}$ for all rooted spiders $\mathcal{X}_j$.

We use the simple algebraic fact that for non-negative numbers a, $x_1, \ldots, x_s$, $y_1, \ldots, y_s$, if $a \leq \frac{x_i}{y_i}$ for all $1 \leq i \leq s$, then $a \leq (\sum_{j=1}^{s} x_j)/(\sum_{j=1}^{s} y_j)$; this is easily verified by writing $ay_i \leq x_i$, then summing over i. Applying this fact over all rooted spiders, we have $\frac{\Delta C_n}{h_n} \leq \frac{\sum_{j=1}^{s} c(\mathcal{X}_j)}{\sum_{j=1}^{s}(1+|L_j|)}$.

Observe that $\sum_{j=1}^{s} c(\mathcal{X}_j) \leq \text{OPT}$, as the vertices in a rooted spider decomposition of G^* are a subset (not necessarily a proper subset) of $V(G^*)$. The denominator, $\sum_{j=1}^{s}(1 + |L_j|)$, equals $|M_n| = |\mathcal{F}_n|$ by Corollary 10. The lemma follows. ◀

We are ready to prove Theorem 2, that GreedyVGSST is a $(2 \ln |T|)$-approximation to the V-GSST problem.

Proof of Theorem 2. Lemma 12 rearranges to $h_n \geq \frac{\Delta C_n}{\text{OPT}} |\mathcal{F}_n|$. Division by zero can occur if $\text{OPT} = 0$, e.g. there is a solution where every vertex v is assigned its minimum grade of service $R(v)$. In this case, GreedyVGSST necessarily returns a solution with zero cost, as $\gamma_n = 0$ on every iteration. Hence, we assume $\text{OPT} > 0$.

Suppose there are I iterations, where $|F_I| \geq 2$ and $|F_{I+1}| := 1$. As $h_n \geq 2$, we equivalently have $\frac{1}{2} h_n \leq h_n - 1$. We have $|\mathcal{F}_{n+1}| \leq |\mathcal{F}_n| - (h_n - 1)$ for each iteration $1 \leq n \leq I$, so Lemma 12 implies the following.

$$
\begin{aligned}
|\mathcal{F}_{n+1}| \le |\mathcal{F}_n| - (h_n - 1) \le |\mathcal{F}_n| - \frac{1}{2}h_n \\
\le |\mathcal{F}_n| \left(1 - \frac{1}{2} \cdot \frac{\Delta C_n}{\mathrm{OPT}}\right)
\end{aligned} \tag{3}
$$

The remainder of the proof relies on unraveling the inequalities and taking the logarithm of both sides.

$$
\begin{aligned}
|\mathcal{F}_{I+1}| &\le |\mathcal{F}_1| \prod_{n=1}^{I} \left(1 - \frac{1}{2} \cdot \frac{\Delta C_n}{\mathrm{OPT}}\right) \\
\ln|\mathcal{F}_{I+1}| &\le \ln|\mathcal{F}_1| + \sum_{n=1}^{I} \ln\left(1 - \frac{1}{2} \cdot \frac{\Delta C_n}{\mathrm{OPT}}\right) \le \ln|\mathcal{F}_1| - \sum_{n=1}^{I} \frac{\Delta C_n}{2 \cdot \mathrm{OPT}}
\end{aligned} \tag{4}
$$

where (4) uses the fact that $\ln(1-x) \le -x$ for $x \in [0,1)$. Note that $0 \le \frac{\Delta C_n}{2 \cdot \mathrm{OPT}} < 1$ as (3) implies $1 - \frac{1}{2} \cdot \frac{\Delta C_n}{\mathrm{OPT}} \ge \frac{|F_{n+1}|}{|F_n|} > 0$. Then (4) implies

$$
\sum_{n=1}^{I} \Delta C_n \le 2 \cdot \mathrm{OPT}(\ln|\mathcal{F}_1| - \ln|\mathcal{F}_{I+1}|) = 2 \cdot \mathrm{OPT}(\ln|T| - \ln 1) = 2\ln|T|\mathrm{OPT}.
$$

completing the proof, as $\sum_{n=1}^{I} \Delta C_n$ is the cost of the solution GreedyVGSST returns. ◀

It is worth noting that the proof of Theorem 2 differs from that of Klein and Ravi [21] in how a rooted spider decomposition in G^* is considered at each iteration. Once the inequality relating h_n with ΔC_n, OPT, and $|\mathcal{F}_n|$ (Lemma 12) is established, the remainder of the proof is similar to that of Klein and Ravi. Note that $|\mathcal{F}_i|$ is strictly decreasing on each iteration, so the number of iterations I is at most $|T| = O(|V|)$. Each iteration can be carried out in polynomial time (Lemma 5), thus GreedyVGSST runs in polynomial time.

In summary, several key techniques allow for the generalization of the KR algorithm [21] to the V-GSST problem. First, merging multiple GRTs is non-trivial, as we must ensure that the resulting tree is also a GRT. Our approach is to connect a root GRT, to a center, to a subset $\mathcal{S}$ of GRTs. This leads to a time complexity increase by a factor of ℓ. Second, the analysis of the KR algorithm [21] relies on the existence of a spider decomposition in the optimal VST solution. For our analysis, we introduce grade-respecting "rooted spiders," characterized by a root, center, and terminal leaves. Third, instead of contracting subtrees computed on iteration i into "supernodes," in the GreedyVGSST algorithm, it is sufficient to only consider distances between GRT roots, and compute the actual tree at the end.

4 Conclusions and future work

We presented a generalization of the VST problem to multiple levels or grades of service and showed that the resulting V-GSST problem admits a $(2\ln|T|)$-approximation, which is surprising as the approximation ratio is optimal (to within a constant), and nearly matches that of the VST problem. The analysis relies on what we call a rooted spider decomposition, which we believe can be of use in other multi-level network design problems. It will be interesting to investigate whether similar generalizations of other graph sparsification problems can be approximated equally as well as their corresponding single-grade problems, and to evaluate the performance of these algorithms on large real-world graphs.

References

1 A. R. Ahmed, P. Angelini, F. Darabi Sahneh, A. Efrat, D. Glickenstein, M. Gronemann, N. Heinsohn, S. Kobourov, R. Spence, J. Watkins, and A. Wolff. Multi-level Steiner trees. In *17th International Symposium on Experimental Algorithms, (SEA)*, pages 15:1–15:14, 2018. doi:10.4230/LIPIcs.SEA.2018.15.

2 A. Balakrishnan, T. L. Magnanti, and P. Mirchandani. Modeling and heuristic worst-case performance analysis of the two-level network design problem. *Management Sci.*, 40(7):846–867, 1994. doi:10.1287/mnsc.40.7.846.

3 M. Bern and P. Plassmann. The Steiner problem with edge lengths 1 and 2. *Inform. Process. Lett.*, 32(4):171–176, 1989. doi:10.1016/0020-0190(89)90039-2.

4 J. Byrka, F. Grandoni, T. Rothvoß, and L. Sanità. Steiner tree approximation via iterative randomized rounding. *J. ACM*, 60(1):6:1–6:33, 2013. doi:10.1145/2432622.2432628.

5 G. Calinescu, S. Kapoor, A. Olshevsky, and A. Zelikovsky. Network lifetime and power assignment in ad hoc wireless networks. In *European Symposium on Algorithms*, pages 114–126. Springer, 2003. doi:10.1007/978-3-540-39658-1_13.

6 M. Charikar, C. Chekuri, T. Cheung, Z. Dai, A. Goel, S. Guha, and M. Li. Approximation Algorithms for Directed Steiner Problems. *J. Algorithms*, 33(1):73 – 91, 1999. doi:10.1006/jagm.1999.1042.

7 M. Charikar, J. Naor, and B. Schieber. Resource optimization in QoS multicast routing of real-time multimedia. *IEEE/ACM Trans. Netw.*, 12(2):340–348, April 2004. doi:10.1109/TNET.2004.826288.

8 C. Chekuri, M. T. Hajiaghayi, G. Kortsarz, and M. R. Salavatipour. Approximation algorithms for node-weighted buy-at-bulk network design. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 1265–1274. SIAM, 2007. doi:10.1145/1283383.1283519.

9 M. Chlebík and J. Chlebíková. The Steiner tree problem on graphs: Inapproximability results. *Theoret. Comput. Sci.*, 406(3):207–214, 2008. doi:10.1016/j.tcs.2008.06.046.

10 J. Chuzhoy, A. Gupta, J. S. Naor, and A. Sinha. On the approximability of some network design problems. *ACM Trans. Algorithms*, 4(2):23:1–23:17, 2008. doi:10.1145/1361192.1361200.

11 R. Cordone and M. Trubian. An exact algorithm for the node weighted steiner tree problem. *4OR*, 4:124–144, 2006. doi:10.1007/s10288-005-0081-y.

12 E. D. Demaine, M. Hajiaghayi, and P. N. Klein. Node-weighted Steiner tree and group Steiner tree in planar graphs. *ACM Trans. Algorithms*, 10(3):13:1–13:20, July 2014. doi:10.1145/2601070.

13 S. Engevall, M. Göthe-Lundgren, and P. Värbrand. A strong lower bound for the node weighted steiner tree problem. *Networks*, 31:11–17, 01 1998. doi:10.1002/(SICI)1097-0037(199801)31:1<11::AID-NET2>3.0.CO;2-N.

14 U. Feige. A threshold of $\ln n$ for approximating set cover. *J. ACM*, 45(4):634–652, 1998. doi:10.1145/237814.237977.

15 E. N. Gilbert and H. O. Pollak. Steiner minimal trees. *SIAM J. Appl. Math.*, 16(1):1–29, 1968. doi:10.1137/0116001.

16 M. X. Goemans and D. P. Williamson. A general approximation technique for constrained forest problems. *SIAM J. Comput.*, 24(2):296–317, 1995.

17 S. Guha and S. Khuller. Improved methods for approximating node weighted Steiner trees and connected dominating sets. *J. Inform. Comput.*, 150(1):57–74, 1999. doi:10.1006/inco.1998.2754.

18 M. Hauptmann and M. Karpinski (eds.). A compendium on Steiner tree problems, 2015. URL: http://theory.cs.uni-bonn.de/info5/steinerkompendium/.

19 R. M. Karp. Reducibility among combinatorial problems. In Raymond E. Miller, James W. Thatcher, and Jean D. Bohlinger, editors, *Complexity of Computer Computations*, pages 85–103. Plenum Press, 1972. doi:10.1007/978-1-4684-2001-2_9.

20 M. Karpinski, I. I. Mandoiu, A. Olshevsky, and A. Zelikovsky. Improved approximation algorithms for the quality of service multicast tree problem. *Algorithmica*, 42(2):109–120, 2005. doi:10.1007/s00453-004-1133-y.

21 P. Klein and R. Ravi. A nearly best-possible approximation algorithm for node-weighted Steiner trees. *J. Algorithms*, 19(1):104–115, 1995. doi:10.1006/jagm.1995.1029.

22 G. Kortsarz and Z. Nutov. Approximating some network design problems with node costs. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, pages 231–243. Springer, 2009. doi:10.1007/978-3-642-03685-9_18.

23 G. R. Mateus, F. R. B. Cruz, and H. P. L. Luna. An algorithm for hierarchical network design. *Location Science*, 2(3):149–164, 1994.

24 P. Mirchandani. The multi-tier tree problem. *INFORMS J. Comput.*, 8(3):202–218, 1996. doi:10.1287/ijoc.8.3.202.

25 A. Moss and Y. Rabani. Approximation algorithms for constrained node weighted Steiner tree problems. *SIAM Journal on Computing*, 37(2):460–481, 2007. doi:10.1137/S0097539702420474.

26 J. Naor, D. Panigrahi, and M. Singh. Online Node-Weighted Steiner Tree and Related Problems. In *2011 IEEE 52nd Annual Symposium on Foundations of Computer Science*, pages 210–219, Oct 2011. doi:10.1109/FOCS.2011.65.

27 Z. Nutov. Approximating minimum power covers of intersecting families and directed edge-connectivity problems. *Theoret. Comput. Sci.*, 411(26-28):2502–2512, 2010. doi:10.1007/11830924_23.

28 Z. Nutov. Approximating Steiner networks with node-weights. *SIAM J. Comput.*, 39(7):3001–3022, 2010. doi:10.1137/080729645.

29 H. J. Prömel and A. Steger. *The Steiner Tree Problem.* Vieweg and Teubner Verlag, 2002. doi:10.1007/978-3-322-80291-0.

30 R. Ravi, M. V. Marathe, S. S. Ravi, D. J. Rosenkrantz, and H. B. Hunt III. Approximation algorithms for degree-constrained minimum-cost network-design problems. *Algorithmica*, 31(1):58–78, 2001. doi:10.1007/s00453-001-0038-2.

31 A. Segev. The Node-Weighted Steiner Tree Problem. *Networks*, 17(1):1–17, February 1987. doi:10.1002/net.3230170102.

32 P. Winter. Steiner problem in networks: A survey. *Networks*, 17(2):129–167, 1987. doi:10.1002/net.3230170203.

33 G. Xue, G. Lin, and D. Du. Grade of service Steiner minimum trees in the Euclidean plane. *Algorithmica*, 31(4):479–500, 2001. doi:10.1007/s00453-001-0050-6.

A V-GSST to DST reduction

The *directed* Steiner tree problem (DST) is often defined as follows: given a directed graph $G = (V, E)$ with edge weights, a set T of terminals, and a root vertex $r \in V$, compute a Steiner arborescence (directed tree) rooted at r, so that there exists an r-v path for each $v \in T$. Segev [31] shows that the VST problem can be transformed into an instance of DST, by replacing each edge uv with two directed edges (u, v), (v, u), where the weight of an edge (u, v) equals the weight of its incoming vertex. Set the root r to be any terminal.

Similarly, we show that the V-GSST problem with arbitrary costs $c_i(v)$ can be formulated as an instance of DST. Given a graph $G = (V, E)$, terminals T with required grades of service $R : T \to \{1, 2, \ldots, \ell\}$, and vertex costs $c_i(v)$, construct ℓ directed copies of G, denoted G^ℓ, $G^{\ell-1}$, …, G^1. Given $v \in V$, let v^i denote the copy of vertex v in G^i. For all $\{u, v\} \in E$, set the costs of directed edges (u^i, v^i) and (v^i, u^i) in G^i to be $c_i(v)$ and $c_i(u)$, respectively. Finally, for all $v \in V$ and $i = 1, 2, \ldots, \ell - 1$, add the directed edge (v^{i+1}, v^i) with cost zero. The interpretation is that if v^i is spanned in a Steiner arborescence, then $v^{i-1}, \ldots, v^1$ are also spanned.

Finally, assign all vertices $v^{R(v)}$ to be terminals in the transformed instance of DST. For the root vertex, select any vertex w with $R(w) = \ell$, and set w^ℓ to be the root. Thus, the transformed instance of DST contains $|V|\ell$ vertices, $2|E|\ell + |V|(\ell - 1)$ directed edges, and $|T|$ terminals. It is not too hard to show that, given an optimal solution to the V-GSST problem, one can construct an equivalent solution to the DST solution with the same cost, and vice versa.

B Proof of Proposition 3

Proof. Here, we can show that $\text{TOP}_i \leq \text{OPT}$ for all $1 \leq i \leq \ell$. Note that, when determining a set of vertices to install grade i facilities on (line 3 in Algorithm 1), a candidate solution is to consider the set of all vertices containing a facility of grade i or greater within the optimal V-GSST G^*, and install facilities of grade i for each vertex in this set. As some vertices may already have facilities of a higher grade installed, the cost incurred when determining E_i is not more than the cost of the facilities of grade i or greater in G^*, which is upper bounded by OPT. Hence $\text{TOP}_i \leq \text{OPT}$. This immediately implies $\text{TOP} \leq \ell \cdot \text{OPT}$. ◀

C Integer linear programming formulation for V-GSST

The following ILP formulation generalizes the cut-based ILP formulation for VST given by Demaine et al. [12].

Given some integer $i \leq \ell$ and $S \subseteq V$, let $f_i(S) = 1$ if at least one terminal in T, but not every terminal, with required grade of service at least i, is in S. Let $f_i(S) = 0$ otherwise. Let $\Gamma(S)$ be the neighborhood of S, defined as the set of vertices adjacent to at least one vertex in S but not in S. For $v \in V$ and $i \in \{1, \ldots, \ell\}$, let x_v^i be a binary indicator variable defined as follows:

$$x_v^i = \begin{cases} 1 & \text{vertex } v \text{ contains a facility of grade } i \text{ or higher} \\ 0 & \text{otherwise} \end{cases}$$

An ILP formulation for the V-GSST problem is as follows:

$$\min \sum_{v\in V}\sum_{i=1}^{\ell}(w_i(v)-w_{i-1}(v))x_v^i \text{ subject to} \tag{5}$$

$$\sum_{v\in\Gamma(S)} x_v^i \geq f_i(S) \qquad \forall v\in V; S\subseteq V \tag{6}$$

$$x_v^i \geq x_v^{i+1} \qquad \forall v\in V; i\in\{1,2,\ldots,\ell-1\} \tag{7}$$

$$x_v^i \in \{0,1\} \qquad \forall v\in V; i\in\{1,2,\ldots,\ell\} \tag{8}$$

The objective (5) splits the cost of installing a facility on v into incremental costs, where $w_0(v)=0$ by definition. Constraint (6) enforces that, for any subset S containing at least one but not all vertices of required grade at least i, the cut is crossed by at least one edge in the solution. Constraint (7) enforces that if a facility is installed on v with grade $i+1$ or higher, then it is installed with grade i, $i-1$, ... or higher. Finally, constraint (8) ensures the x_v^i's are binary.

Such an ILP will only output the x_v^i's, though one can easily extract the assigned grades of service $y(v)$ and construct a valid V-GSST solution given this information.