

Computing a Minimum-Cost k -Hop Steiner Tree in Tree-Like Metrics

Martin Böhm 

University of Bremen, Germany
martin.boehm@uni-bremen.de

Ruben Hoeksma 


University of Twente, The Netherlands
r.p.hoeksma@utwente.nl

Nicole Megow 

University of Bremen, Germany
nicole.megow@uni-bremen.de

Lukas Nölke 

University of Bremen, Germany
noelke@uni-bremen.de

Bertrand Simon 

University of Bremen, Germany
bsimon@uni-bremen.de

Abstract

We consider the problem of computing a Steiner tree of minimum cost under a k -hop constraint which requires the depth of the tree to be at most k . Our main result is an exact algorithm for metrics induced by graphs of bounded treewidth that runs in time $n^{O(k)}$. For the special case of a path, we give a simple algorithm that solves the problem in polynomial time, even if k is part of the input. The main result can be used to obtain, in quasi-polynomial time, a near-optimal solution that violates the k -hop constraint by at most one hop for more general metrics induced by graphs of bounded highway dimension and bounded doubling dimension.

2012 ACM Subject Classification Theory of computation → Graph algorithms analysis; Mathematics of computing → Combinatorial optimization

Keywords and phrases k -hop Steiner tree, dynamic programming, bounded treewidth

Digital Object Identifier 10.4230/LIPIcs.MFCS.2020.18

Related Version A full version of the paper is available at <https://arxiv.org/abs/2003.05699>.

Funding Partially funded and supported by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) Projects ME 3825/1 and 146371743 – TRR 89 Invasive Computing.

Acknowledgements We thank Jiří Sgall for discussions on k -hop MST on paths and an anonymous reviewer for pointing us to the connection between highway dimension and doubling dimension.

1 Introduction

We are given a finite metric space (V, d) with a set of n points V and a distance function $d : V \times V \rightarrow \mathbb{Q}_+$, a set of terminals $\mathcal{X} \subseteq V$, a root $r \in \mathcal{X}$, and an integer $k \geq 1$. A k -hop Steiner tree is a tree $\check{S} = (V_{\check{S}}, E_{\check{S}})$ rooted at r that spans all points in \mathcal{X} and has a depth of at most k . That is, $\mathcal{X} \subseteq V_{\check{S}} \subseteq V$ and for $v \in V_{\check{S}}$, the number of edges in the r - v path in \check{S} is at most k . The cost of a Steiner tree is the sum of edge costs $\sum_{\{u,v\} \in E_{\check{S}}} d(u,v)$, with



© Martin Böhm, Ruben Hoeksma, Nicole Megow, Lukas Nölke, and Bertrand Simon; licensed under Creative Commons License CC-BY

45th International Symposium on Mathematical Foundations of Computer Science (MFCS 2020).

Editors: Javier Esparza and Daniel Král'; Article No. 18; pp. 18:1–18:15

Leibniz International Proceedings in Informatics



LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

edge costs given by d . We consider the *minimum-cost k -hop Steiner tree problem* (k -hop MŠT problem¹) that asks for a k -hop Steiner tree of minimum cost. When $\mathcal{X} = V$, this is equivalent to the *minimum-cost k -hop spanning tree* (k -hop MST) problem.

The k -hop MŠT problem is highly relevant for many applications, e.g., in the design of transportation and communication networks, particularly regarding the efficiency and reliability of routing. A restriction on the hop distances aims at reducing transmission delays, avoids flooding the network when routing, reduces packet loss and increases reliability of networks by limiting the amplifying effect of link failures. There exists a multitude of applications; see, e.g., [7, 11, 15, 18, 19, 22, 31, 32].

In this work, we show how to solve the k -hop MŠT problem in certain tree-like metrics. That is, we consider metrics which are represented by graphs from certain tree-like graph classes using the natural correspondence between metric spaces and weighted complete graphs via the shortest path metric. We say a weighted graph $G = (V, E)$ induces a metric (V, d) if for any two vertices $u, v \in V$ the length of the shortest u - v path in G equals $d(u, v)$. A metric is called a *tree* (resp. *path*) *metric* if there is a tree (resp. path) inducing it, and it is called a *metric of bounded treewidth* if it is induced by some graph with bounded treewidth. For a given metric, it can be decided in polynomial time if it is a path metric, a tree metric, or a metric of constant treewidth ω ; details are outlined in Section 2. For convenience, we may not always distinguish between a metric and the graph inducing it.

Previous work. Hop-constrained problems have been studied since the 1980s. Various well-studied problems are in fact special cases of the k -hop MŠT problem, most notably, the k -hop MST problem, where $\mathcal{X} = V$, the Minimum Steiner Tree problem, where $k \geq n$, and the Uncapacitated Facility Location problem, where $k = 2$. Hardness and inapproximability results are therefore valid for k -hop MŠT as well. In particular, k -hop MŠT is NP-hard [4], even for graph metrics, while the Minimum Steiner Tree problem is polynomial-time solvable on graphs of bounded treewidth [12].

When considering metrics more general than those of bounded treewidth, several hardness results are known. Bern and Plassmann [9] show that the Steiner tree problem on a metric induced by a complete graph with edge weights 1 or 2 is MaxSNP-hard. The same is shown for metric 2-hop MST by Alfandari and Paschos [3]. Thus, these problems do not admit a PTAS, unless $P = NP$. Manyem and Stallmann [30] show that k -hop MŠT on a graph with unit-weight edges and 2-hop MST cannot admit a constant-factor approximation algorithm. They also show that k -hop MST on a graph with edge weights 1 or 2 cannot admit a PTAS. For general non-metric graphs, Alfandari and Paschos [3] prove that even for 2-hop MST no $(1 - \varepsilon) \log(n)$ -approximation can exist unless $NP \subseteq DTIME[n^{O(\log \log n)}]$.

The following works, while conceptually closest to our paper, focus on approximation algorithms. Kortsarz and Peleg [27] consider k -hop MŠT on non-metric graphs obtaining a approximation factor $O(\log n)$ for constant k and $O(n^\varepsilon)$ otherwise. Althaus et al. [4] give a $O(\log n)$ -approximation for arbitrary k for metric k -hop MST that first uses a randomized embedding of the given metric into a hierarchically-separated tree and then solves this problem optimally. For constant k , Laue and Matijević [28] derive a PTAS for k -hop MŠT in the plane. Their algorithm implies a QPTAS for Euclidean spaces of higher dimensions. While the first constant-factor approximation algorithm for metric k -hop MŠT is due to

¹ For brevity and as homage to the work of Jarník and Kössler [24, 26], we use the Czech letter Š to distinguish Steiner trees from spanning trees in MŠT resp. MST. The pronunciation of Š is (sh), the same as the German pronunciation of the letter S in Steiner.

Kantor and Peleg [25], the factor $1.52 \cdot 9^{k-2}$ is prohibitively high. For $k = 2$, a nearly optimal algorithm is known. The best known approximation ratio of 1.488 for metric Uncapacitated Facility Location [29] and lower bound of 1.463 [20] are valid for metric 2-hop MST as well.

The bounded-diameter minimum Steiner tree problem [19, 25] is also closely related to our bounded-hop problem, yet neither a generalization, nor a special case. Here, for given d we look for a minimum-cost Steiner tree with diameter at most d . For constant d , an $O(1)$ -approximation algorithm is known for graph metrics [25]. For non-metric cost functions, an $o(\log n)$ -approximation algorithm has been ruled out, assuming $P \neq NP$ [8].

Furthermore, shallow-light and buy-at-bulk Steiner trees [6, 13, 16, 23, 27] are conceptually similar to k -hop MŠTs. However, a key difference is that, here, lengths of paths in the tree are bounded w.r.t. metric distance instead of the number of edges on the path. Elkin and Solomon [16] additionally bound the number of hops, but do so by $O(\log n)$ to bound other measures of interest. Chimani and Spoerhase [13] consider two different measures for distance and weight and achieve an n^ε -approximation, violating the distance by a factor of $1 + \varepsilon$.

Minimum-cost k -hop spanning and Steiner trees have been studied in the context of random graphs as well. There, the goal is to give estimates on the weight of an optimal tree. In this setting, sharp threshold for k are known [5].

Our Results. In Section 3, we give a quite simple exact algorithm for the path metric which runs in polynomial time, even when k is part of the input.

► **Theorem 1.** *On path metrics, k -hop MŠT can be solved exactly in time $O(kn^5)$.*

Our main result is a dynamic program (DP) for metrics with bounded treewidth. In Section 4, we first consider the special case of tree metrics. Here, cells are indexed by a vertex v as well as $2k$ additional vertices. The latter represent possible parents of v at different depths in a k -hop MŠT. Specifically, for each depth in this Steiner tree, there is one possible parent in $T[v]$ and one outside, where $T[v]$ denotes the subtree (w.r.t. the tree metric) rooted at v .

Our DP is substantially different from that in [4] which is tailored to hierarchically-separated trees. While the DP for planar graphs in [28] has similarities to our construction for tree metrics, a notable difference lies in the indexing of their cells by distances. In our case, such a strategy does not carry enough information; hence, we resort to indexing by vertices, as explained above, and retain more structure.

In Section 5, we extend the approach to metrics of bounded treewidth.

► **Theorem 2.** *On metrics of treewidth ω , k -hop MŠT can be solved exactly in time $n^{O(\omega k)}$.*

This result also facilitates a quasi-polynomial time approximation algorithm for more general metrics induced by graphs of bounded highway dimension. This graph class was introduced in [2] to model transportation networks. Intuitively, in graphs of bounded highway dimension, locally, there exists a small set of transit vertices such that the shortest paths between two distant vertices pass through some transit vertex; details in Section 6.

Using a framework from [17], we show in Section 6 that Theorem 2 and the constant-factor approximation designed in [25] lead to the following result.

► **Theorem 3.** *For a metric induced by a graph of bounded highway dimension and a constant k , let OPT_k be the cost of a k -hop MŠT. A $(k + 1)$ -hop Steiner tree of cost at most $(1 + \varepsilon)OPT_k$, for $\varepsilon > 0$, can be computed in quasi-polynomial time.*

This seems to be the first result with resource augmentation in the context of hop-constrained network design. This research direction was proposed in [4].

There is a close relation between the highway dimension and the well-known concept of doubling dimension [21]. A metric space has doubling dimension d if every ball of radius r can be covered by $2d$ balls of radius $r/2$. Abraham et al. [1] discuss the relation between these two concepts and show that constant doubling dimension does not imply constant highway dimension but that the converse is true. Hence, Theorem 3 directly implies the following.

► **Corollary 4.** *For a metric of bounded doubling dimension and a constant k , let OPT_k be the cost of a k -hop MŠT. A $(k+1)$ -hop Steiner tree of cost at most $(1+\varepsilon)OPT_k$, for $\varepsilon > 0$, can be computed in quasi-polynomial time.*

2 Preliminaries

Let (V, d) be a metric induced by the graph $G = (V, E)$. We assume that G is the *minimal graph inducing* (V, d) , that is, there is no edge in G that can be removed without changing some shortest path. In order to break ties consistently, we assume shortest paths in G to be unique. This can be achieved by adding some sufficiently small noise to the input by slightly moving each point. A k -hop MŠT for the modified instance is also optimal for the original instance. Furthermore, the minimal graph G inducing (V, d) is unique.

Given a metric (resp. the inducing graph), we can decide in polynomial time whether it is a path metric, a tree metric, or a metric of treewidth ω for some constant $\omega \geq 1$. To do so, we determine G efficiently by computing all-pair shortest paths and keeping only those edges in G that are part of some shortest path. To verify that G is a path or a tree, we simply run a depth-first search. Moreover, for constant ω , it can be decided in polynomial time whether G has treewidth ω by computing a treewidth decomposition [10].

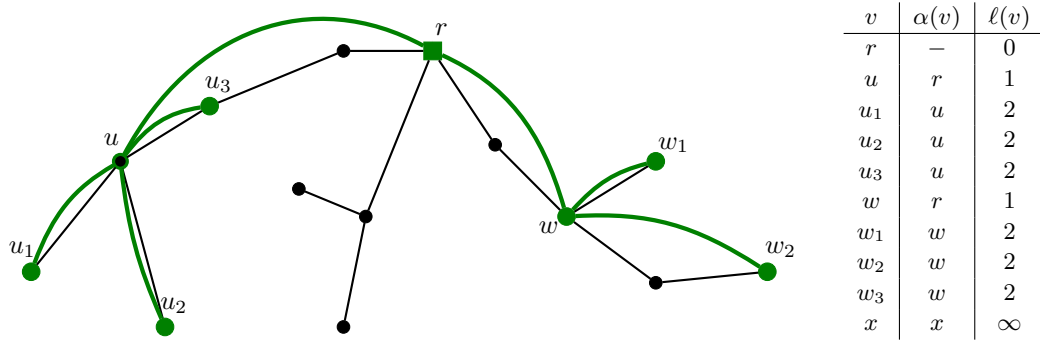
We give two alternative representations of Steiner trees that are useful when working with partial solutions. Let \check{S} be a Steiner tree on (V, d) with terminals $\mathcal{X} \subseteq V$ and root $r \in \mathcal{X}$. Let $V_{\check{S}} \subseteq V$ with $\mathcal{X} \subseteq V_{\check{S}}$ be the set of vertices in \check{S} . The tree \check{S} can be viewed as a function mapping a vertex of $V_{\check{S}} \setminus \{r\}$ to its immediate predecessor, i.e., its parent in \check{S} . More generally, for $U \subseteq V$, call a function $\alpha : U \setminus \{r\} \rightarrow V$ an *anchoring on U* . The *anchor* $\alpha(v)$ of vertex v represents its parent in \check{S} , and we set $\alpha(w) = w$ if $w \notin V_{\check{S}}$.

If \check{S} is of minimum cost, this additionally allows for the following representation. Consider a function assigning to each vertex $v \in V_{\check{S}}$ its depth, i.e. the number of edges on the r - v path in \check{S} . Since a vertex $v \in V_{\check{S}} \setminus \{r\}$ of depth x is anchored to the (uniquely determined) vertex of depth $x-1$ that is of minimum distance to v w.r.t. d , this yields a complete representation of \check{S} . Generalizing again to subsets $U \subseteq V$, we call a function $\ell : U \rightarrow \{0, 1, \dots, k\} \cup \{\infty\}$ a *labeling on U* . We call $\ell(w)$ the *label of w* and set $\ell(w) = \infty$ if $w \notin V_{\check{S}}$. Note that this representation automatically enforces the k -hop condition. See Figure 1 for an example of a k -hop MŠT with the corresponding anchoring and labeling.

When \check{S} is of minimum cost and $U = V$, we can easily compute an anchoring from a labeling or vice versa. However, when considering partial solutions, i.e., when $U \subsetneq V$, this may not be possible. Thus, to retain the essential structural information, we utilize both representations simultaneously in this case. This motivates the following definition.

► **Definition 5.** *A pair (ℓ, α) is called a labeling-anchoring pair (LAP) on U if the labeling ℓ and anchoring α are consistent, i.e. for every $u \in U \setminus \{r\}$ for which $\alpha(u) \in U$ and $\ell(u) \neq \infty$, we have $\ell(u) = \ell(\alpha(u)) + 1$. Moreover, if $\ell(u) = \infty$ then $u \notin \mathcal{X}$ and $\alpha^{-1}(u) = \{u\}$.*

The cost of a LAP (ℓ, α) is given by $\sum_{u \in U \setminus \{r\}} d(u, \alpha(u))$. In this sum, the term $d(u, \alpha(u))$ is called the *cost to anchor u* . When $U \subsetneq V$, we may say *partial LAP* to emphasize that the LAP only represents a portion of \check{S} , namely the edges between U and its anchors.



■ **Figure 1** A 2-hop MŠT (green) with root r and terminals $u_1, u_2, u_3, r, w, w_1, w_2$ on a tree metric (black) with unit-weight edges. Its cost is 11. The table on the right describes the corresponding labeling ℓ and anchoring α , where x symbolizes vertices not used by the MŠT.

Furthermore, we use the representation as LAP to avoid the ambiguity that arises from simultaneously considering a Steiner tree \check{S} and the tree-like graph G that induces the underlying metric space. For example, in Section 4, both \check{S} and G are trees. Throughout the paper, we represent Steiner trees as LAPs. Hence, we use the term *anchor* to refer to a predecessor in \check{S} instead of *parent*. Additionally, when talking about distances or closeness, we always refer to distances in G . Given a point v and a set $U \subseteq V$, denote by $\text{closest}_v(U)$ the (unique) element of U with minimum distance to v . For simplicity, we write $\text{closest}_v(u, w)$ instead of $\text{closest}_v(\{u, w\})$.

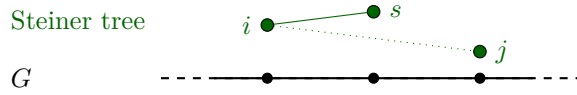
In Sections 4 and 5, when querying a DP cell, a vertex with a desired label may not exist. To make these queries technically simple, we extend the vertex set of the metric to contain an auxiliary vertex, denoted by v_\emptyset . It is defined to have distance ∞ to all other vertices. In order to avoid the use of k auxiliary vertices (one per label), we slightly abuse notation and assume that the equality $\ell(v_\emptyset) = i$ is correct for all $i \in [k]$ where $[k] = \{1, 2, \dots, k\}$. Note that anchoring v_\emptyset incurs an infinite cost, so it will never be used in a k -hop Steiner tree.

3 The k -hop MŠT Problem in Path Metrics

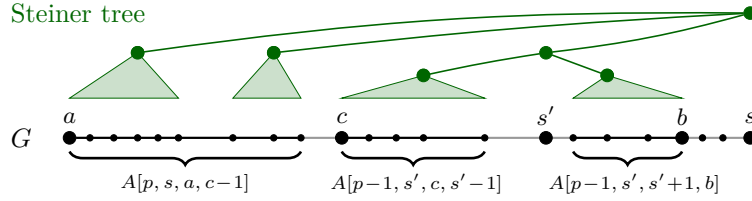
Our first result is an efficient algorithm for k -hop MŠT on path metrics. We view a path metric as a set of vertices $V = \{v_1, v_2, \dots, v_n\}$ placed on the real line from left to right, such that edges in the path correspond to consecutive vertices. In this special case, there exists a (uniquely defined) minimum-cost k -hop MŠT $\text{OPT} = (\ell, \alpha)$ rooted at $r \in V$ that only uses terminals. Indeed, if OPT contains a non-terminal vertex v , we may simply replace it by the next vertex on the line in the direction in which v has the most edges (break ties arbitrarily). This removes a non-terminal vertex without increasing the cost of OPT or violating the k -hop condition. In this section, we therefore assume $\mathcal{X} = V$.

We give a recursive procedure which computes the k -hop MST, and discuss the complexity of computing it via dynamic programming. The goal is to first compute the internal (non-leaf) vertices of the k -hop MST, and then add the cost of anchoring the leaves to the closest internal vertices.

A key observation is the following. Fix an internal vertex s of depth $\ell(s) < k$. It partitions the remaining vertex set into the vertices on the left of s , and those on the right of s . If a vertex i to the left of s is of depth $\ell(i) > \ell(s)$, then in OPT , the vertex i is never adjacent to a vertex to the right of s , see Figure 2. This follows from the fact that such a vertex could be attached to s directly, decreasing the overall cost of OPT without using more hops.



■ **Figure 2** The optimal k -hop MŠT never attaches j to i if $\ell(i) > \ell(s)$.



■ **Figure 3** Computation of $A[p, s, a, b]$ with three recursive calls.

We define a recursive expression $A[p, s, a, b]$ for $p \in \mathbb{N}$ and $s, a, b \in [n]$. Intuitively, it yields the minimum cost p -hop spanning tree \check{S} rooted at v_s that contains all vertices v_i with $i \in [a, b]$ and satisfies $s \notin [a, b]$. If $a > b$, let $[a, b] = \emptyset$.

For $p \in \mathbb{N}$ and $s, a, b \in [n]$, define $A[p, s, a, b]$ as follows.

1. If $a > b$, then $A[p, s, a, b] = 0$.
2. If $a = b$, then $A[p, s, a, a] = d(v_s, v_a)$.
3. If $p = 1$, then $A[1, s, a, b] = \sum_{x \in [a, b]} d(v_s, v_x)$ (all vertices anchored to v_s).
4. If $p > 1$, consider the right-most child $v_{s'}$ of v_s in \check{S} such that $s' \in [a, b]$. The sub-tree of \check{S} rooted at $v_{s'}$ covers all vertices v_i with $i \in [c, b]$ for some $c \in [a, s']$. Thus $A[p, s, a, b]$ is the sum of the cost of this subtree and that of all remaining subtrees of v_s in $[a, c - 1]$. That is, $A[p, s, a, b]$ is defined as

$$\min_{s' \in [a, b], c \in [a, s'-1]} d(v_{s'}, v_s) + A[p, s, a, c - 1] + A[p - 1, s', c, s' - 1] + A[p - 1, s', s' + 1, b].$$

See Figure 3 for an illustration where $b < s$. Note that in the last case, any recursive call can refer to an empty interval and incur zero cost.

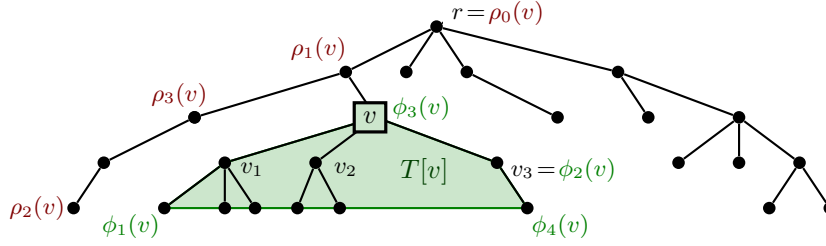
Proof of Theorem 1. Due to the key observation above, $A[p, s, a, b]$ correctly computes the minimum cost of a p -hop spanning tree \check{S} with root v_s and vertices v_i with $i \in [a, b]$: For s' and c as in OPT, there are no edges in OPT between $[a, c - 1]$, $[c, s' - 1]$ and $[s' + 1, b]$. Also, the recursive procedure only queries intervals $[a, b]$ with $s \notin [a, b]$. The cost of OPT is $A[k, r, 0, r - 1] + A[k, r, r + 1, n]$.

We dynamically compute the values $A[p, s, a, b]$ by iterating in an increasing manner over p in an outer loop and the set of intervals $[a, b]$ in an inner loop, with shorter intervals having precedence. This is feasible, as a call of $A[p, s, a, b]$ recursively only queries values $A[p', s', a', b']$ with $p' < p$ or $(b' - a')^+ < (b - a)^+$. Assuming that all previous values are precomputed, the value of a cell $A[p, s, a, b]$ can be computed in time $O(n^2)$. Since there are only kn^3 possible values of (p, s, a, b) to be queried, the total running time is bounded by $O(kn^5)$. ◀

4 The k -hop MŠT Problem in Tree Metrics

► **Theorem 6.** *In tree metrics, k -hop MŠT can be solved exactly in time $n^{O(k)}$.*

In this section, we construct a dynamic program for the k -hop MŠT problem on tree metrics. Consider an instance of k -hop MŠT with root $r \in \mathcal{X}$ and metric (V, d) induced by a tree $T = (V, E)$. Without loss of generality, we consider T to be rooted at r . For $v \in V$, denote by $T[v]$ the set of vertices in the subtree of T rooted at v .



■ **Figure 4** Possible anchoring guarantees $\rho(v)$, $\phi(v)$ for vertex v . Its subtree $T[v]$ with respect to the underlying metric (black) is highlighted in green. To satisfy the anchoring guarantees, in a Steiner tree, v must be anchored to either $\phi_2(v)$ or $\rho_2(v)$. Note that $k = 4$ and $\rho_4(v) = v_\emptyset$.

We start by giving a high-level overview of our approach for computing the minimum cost k -hop Steiner tree $\text{OPT} = (\ell, \alpha)$. We use a dynamic program with cells $\bar{A}[v, \rho, \phi]$ indexed by a node $v \in V$ and vectors ρ and ϕ of k vertices each. Intuitively, ρ and ϕ represent *anchoring guarantees* that convey information about the structure of OPT in relation to v and serve as possible points to which v is anchored in α . Specifically, for each possible label i , there are two anchoring guarantees $\phi_i \in T[v]$ and $\rho_i \in V \setminus T[v]$ with $\ell(\phi_i) = \ell(\rho_i) = i$ that act as candidates for anchoring v in OPT to a vertex of depth i . If $\ell(v) = i + 1$, then $\alpha(v) = \text{closest}_v(\phi_i, \rho_i)$. We show that a cell $\bar{A}[v, \rho, \phi]$ computes a partial labeling-anchoring pair (LAP, recall Definition 5) on $T[v]$ that is of minimum cost and respects the given anchoring guarantees. The cells are filled up in a bottom-to-top manner, starting at the leaves of the underlying tree T . Doing this consistently, while filling in correct anchoring guarantees, finally yields OPT .

Anchoring guarantees. Fix a vertex $v \in V \setminus \{r\}$. Formally, its *anchoring guarantees* are given by $\phi(v) = (\phi_1(v), \dots, \phi_{k-1}(v))$ and $\rho(v) = (\rho_1(v), \dots, \rho_{k-1}(v))$ such that $\phi_i(v) \in T[v]$ and $\rho_i(v) \in V \setminus T[v]$ for all $i \in [k-1]$. Additionally, we allow the $\phi_i(v)$ and $\rho_i(v)$ to take the value v_\emptyset and let $\rho_0(v) = r$ and $\phi_0(v) = v_\emptyset$; see Figure 4.

In our search for partial solutions, we are interested in partial LAPs on $T[v]$. Given a LAP, denote by $\lambda_i(v)$ the vertex in $T[v]$ of label i closest to v (or v_\emptyset if no such vertex exists). Let $\mathcal{P}(v, \rho(v), \phi(v))$ be the (possibly empty) set of LAPs on $T[v]$ respecting the anchoring guarantees. That is, its elements (ℓ, α) satisfy:

- (i) For all i , we have $\phi_i(v) = \lambda_i(v)$. In particular, if $\phi_i(v) = \phi_j(v)$ and $i \neq j$, then $\phi_i(v) = v_\emptyset$.
- (ii) A vertex $w \in T[v]$ with $\ell(w) \neq \infty$ is anchored to a vertex of $T[v]$ with label $\ell(w) - 1$ or to $\rho_{\ell(w)-1}(v)$. Recall that $\ell(w) = \infty$ implies $\alpha(w) = w$ (and $w \notin \mathcal{X}$).

Intuitively, $\mathcal{P}(v, \rho(v), \phi(v))$ represents all relevant ways to extend a partial LAP (ℓ', α') on $V \setminus T[v]$ to V while respecting the anchoring guarantees. Note that vertices of $T[v]$ are anchored either to another vertex in $T[v]$ or to some $\rho_i(v)$. Therefore, if $\rho_i(v)$ is used, it should be the closest vertex to v outside of $T[v]$ for which $\ell'(\rho_i(v)) = i$. Assume (ℓ', α') is extended with minimum cost and consider the subtree $T[v_j]$ of a child v_j of v . Its vertices are anchored either to a vertex of $T[v_j]$, or to a $\phi_i(v)$ (which may be in the subtree of a different child), or to a $\rho_i(v)$. The anchoring guarantees $\phi_i(v)$ are then necessary to determine the anchoring guarantees $\rho_i(v_j)$ for the children of v . Note that when defining \mathcal{P} , we do not require any constraint on the values of ρ .

The dynamic program. For $v \neq r$, let $A[v, \rho(v), \phi(v)]$ be the minimum cost of a LAP on $T[v]$ in $\mathcal{P}(v, \rho(v), \phi(v))$, or ∞ if none exists. Denote by v_1, v_2, \dots, v_p the children of v in T . We fill the cells $\bar{A}[v, \rho(v), \phi(v)]$ of our dynamic programming table according to the

following recursive relation. For a vertex v that is a leaf of T , we define

$$\bar{A}[v, \rho(v), \phi(v)] := \begin{cases} 0, & \text{if } v \notin \mathcal{X} \text{ and } \phi_i(v) = v_\emptyset \text{ for all } i; \\ d(v, \rho_{i_v-1}(v)), & \text{if } \exists \text{ unique } i_v, \text{ s.t. } \phi_{i_v}(v) = v; \\ \infty, & \text{otherwise.} \end{cases} \quad (1)$$

For non-leaf vertices, we define

$$\bar{A}[v, \rho(v), \phi(v)] := c_v + \sum_{j=1}^p \min_{\phi_i(v_j) \in \Phi_i(v_j), \forall i} \bar{A}[v_j, \rho(v_j), \phi(v_j)]. \quad (2)$$

Here, c_v is the cost of anchoring v while $\Phi_i(v_j)$ and $\rho(v_j)$ encode which of the n^{2k-2} possible anchoring guarantees of v_j are consistent with that of v . The cells of each child are queried independently. Precise definitions of $\Phi_i(v_j)$, $\rho(v_j)$ and c_v follow.

Let $\Phi_i(v_j)$ be the subset of $T[v_j]$ consisting of all feasible choices for $\phi_i(v_j)$. Specifically, if $\phi_i(v) \in T[v_j]$, then $\Phi_i(v_j) = \{\phi_i(v)\}$. Indeed, as the shortest v - $\phi_i(v)$ path passes through v_j , node $\phi_i(v)$ must be the closest vertex to v_j in $T[v_j]$ with (already guaranteed) label i . If $\phi_i(v) = v_\emptyset$, we must have $\Phi_i(v_j) = \{v_\emptyset\}$ or contradict Property (i). Otherwise, if $v_\emptyset \neq \phi_i(v) \notin T[v_j]$, then $\Phi_i(v_j)$ contains all $w \in T[v_j]$ with $d(v, w) \geq d(v, \phi_i(v))$ and the auxiliary vertex v_\emptyset . A distance $d(v, w) < d(v, \phi_i(v))$ would contradict the choice of $\phi_i(v)$ as the vertex in $T[v]$ of label i closest to v .

As for $\rho_i(v_j)$, we define it to be the feasible choice for $\rho_i(v_j)$, which is (uniquely) determined as follows. If $\phi_i(v) \in T[v_j]$, then $\rho_i(v_j) = \rho_i(v)$ since the shortest v_j - $\rho_i(v_j)$ path passes through v . Otherwise, we have $\rho_i(v_j) = \text{closest}_v(\rho_i(v), \phi_i(v))$.

We now define c_v . If $v \notin \mathcal{X}$ and no $\phi_i(v)$ equals v , then $c_v := 0$. Next, if there exists a unique i_v such that $\phi_{i_v}(v) = v$, let $c_v := d(v, \text{closest}_v(\rho_{i_v-1}(v), \phi_{i_v-1}(v)))$. In all other cases set $c_v := \infty$, as the values of $\phi(v)$ are contradictory.

Proof of Theorem 6. By mathematical induction, we prove that $\bar{A}[v, \rho(v), \phi(v)]$, as defined in Equations (1) and (2), is equal to $A[v, \rho(v), \phi(v)]$, for $v \neq r$ and every $\rho(v)$, $\phi(v)$.

For the base step, i.e. when v is a leaf of T , we consider the three cases of Equation (1). If $v \notin \mathcal{X}$ and $\phi_i(v) = v_\emptyset$ for all i , then clearly Properties (i) and (ii) are satisfied for the LAP that excludes v from the Steiner tree, so $\bar{A}[v, \rho(v), \phi(v)] = A[v, \rho(v), \phi(v)] = 0$. Otherwise, there is at most one LAP that satisfies (i) and (ii), namely the one that anchors v to $\rho_{i_v-1}(v)$ if i_v is defined. It incurs a cost of $d(v, \rho_{i_v-1}(v))$, as desired. If no such LAP exists, $\mathcal{P}(v, \rho(v), \phi(v)) = \emptyset$ and both $A[v, \rho(v), \phi(v)]$ and $\bar{A}[v, \rho(v), \phi(v)]$ are infinite. This concludes the base step.

Our induction hypothesis is that

$$\bar{A}[v', \rho(v'), \phi(v')] = A[v', \rho(v'), \phi(v')], \quad \text{for all } v' \neq r, \rho(v'), \text{ and } \phi(v'). \quad (\text{IH})$$

Now, for some non-leaf v , we assume that (IH) holds for all children $v' \in T[v] \setminus \{v\}$ of v and prove that (IH) holds for v as well. For v , the recursive equation (2) becomes

$$\bar{A}[v, \rho(v), \phi(v)] := c_v + \sum_{j=1}^p \min_{\phi_i(v_j) \in \Phi_i(v_j), \forall i} \bar{A}[v_j, \rho(v_j), \phi(v_j)].$$

If $c_v = \infty$, then $\mathcal{P}(v, \rho(v), \phi(v)) = \emptyset$, so both $A[v, \rho(v), \phi(v)] = \infty = \bar{A}[v, \rho(v), \phi(v)]$. From now on, assume that c_v is finite. We prove (IH) for v by showing the two inequalities $A[v, \rho(v), \phi(v)] \geq \bar{A}[v, \rho(v), \phi(v)]$ and $A[v, \rho(v), \phi(v)] \leq \bar{A}[v, \rho(v), \phi(v)]$.

First direction, $A[v, \rho(v), \phi(v)] \geq \bar{A}[v, \rho(v), \phi(v)]$. Consider the LAP (ℓ, α) which yields the value $A[v, \rho(v), \phi(v)]$. In particular, Properties (i) and (ii) are satisfied. If no such LAP exists, then $A[v, \rho(v), \phi(v)] = \infty$ and the inequality holds. For each child v_j of v , set $\phi_i(v_j) = \lambda_i(v_j)$, which respects $\phi_i(v_j) \in \Phi_i(v_j)$. Also, set $\rho(v_j) = \rho(v_j)$ as defined above. We show that for each v_j , the restriction of the LAP (ℓ, α) to $T[v_j]$ belongs to $\mathcal{P}(v_j, \rho(v_j), \phi(v_j))$.

Property (i) follows directly from the choice of $\phi_i(v_j) = \lambda_i(v_j)$.

For Property (ii), consider a vertex $w \in T[v_j]$ which is not anchored to a vertex of $T[v_j]$. We show that α anchors w to $\rho_{\ell_w}(v_j)$, with $\ell_w := \ell(w) - 1$. Note that by definition of $\rho(v_j)$, we have that $\rho_{\ell_w}(v_j)$ is equal to $\rho_{\ell_w}(v)$ or $\phi_{\ell_w}(v)$, so $\ell(\rho_{\ell_w}(v_j)) = \ell_w$. Since α is an anchoring of minimal cost (with respect to the given guarantees), w is anchored to the vertex $\alpha(w) = \text{closest}_w \{x \in T[v] \cup \{\rho_{\ell_w}(v_j)\} \mid \ell(x) = \ell_w\}$, so $\alpha(w) = \text{closest}_{v_j}(\rho_{\ell_w}(v), \phi_{\ell_w}(v))$. If $\phi_{\ell_w}(v) \in T[v_j]$, then $\rho_{\ell_w}(v_j) = \rho_{\ell_w}(v) = \alpha(w)$ since w is not anchored to a vertex in $T[v_j]$. If $\phi_{\ell_w}(v) \notin T[v_j]$, then $\rho_{\ell_w}(v_j) = \text{closest}_{v_j}(\rho_{\ell_w}(v), \phi_{\ell_w}(v)) = \alpha(w)$, by definition of $\rho(v_j)$.

Therefore, the LAP (ℓ, α) restricted to $T[v_j]$ belongs to $\mathcal{P}(v_j, \rho(v_j), \phi(v_j))$, so its cost is at least $A[v_j, \rho(v_j), \phi(v_j)]$. If $\ell(v) \neq \infty$, then $\alpha(v) = \text{closest}_v(\rho_{i_v-1}(v), \phi_{i_v-1}(v))$ with cost c_v , since the anchoring cost is minimized. If $\ell(v) = \infty$, then $c_v = 0$, so

$$A[v, \rho(v), \phi(v)] = c_v + \sum_{j=1}^p A[v_j, \rho(v_j), \phi(v_j)] \geq \bar{A}[v, \rho(v), \phi(v)].$$

Second direction, $A[v, \rho(v), \phi(v)] \leq \bar{A}[v, \rho(v), \phi(v)]$. We assume $\bar{A}[v, \rho(v), \phi(v)]$ to be finite, otherwise the inequality trivially holds. Consider the LAPs corresponding to the values $A[v_j, \rho(v_j), \phi(v_j)]$ for which the value $\bar{A}[v, \rho(v), \phi(v)]$ is attained. We extend these LAPs on the subtrees $T[v_j]$ to (ℓ, α) on $T[v]$ in the following way. If $v \notin \mathcal{X}$ and no $\phi_i(v)$ equals v , we let $\ell(v) = \infty$ and $\alpha(v) = v$. Otherwise, as $c_v \neq \infty$ by our assumption at the start of the proof, there exists a unique i_v such that $\phi_{i_v}(v) = v$. We then let $\ell(v) = i_v$ and anchor v to $\text{closest}_v(\rho_{i_v-1}(v), \phi_{i_v-1}(v))$. We show that this yields an element of $\mathcal{P}(v, \rho(v), \phi(v))$.

We first show Property (i). If i_v is defined, $\phi_{i_v}(v) = v = \lambda_{i_v}(v)$ since $\ell(v) = i_v$. Consider $\phi_i(v)$ for $i \neq i_v$. If $\phi_i(v) = v_\emptyset$, then all $\phi_i(v_j) = v_\emptyset$ too by definition of $\Phi_i(v_j)$. Thus, $\lambda_i(v_j) = v_\emptyset$ for all j and $\ell(v) \neq i$, so $\lambda_i(v) = v_\emptyset = \phi_i(v)$. Otherwise, if $\phi_i(v) \neq v_\emptyset$, there exists a j_i with $\phi_i(v) \in T[v_{j_i}]$. Then, we have $\lambda_i(v_{j_i}) = \phi_i(v)$, and for all j , we have $d(v, \lambda_i(v_j)) = d(v, \phi_i(v_{j_i})) \geq d(v, \phi_i(v))$. Since $\ell(v) \neq i$, we obtain $\lambda_i(v) = \phi_i(v)$.

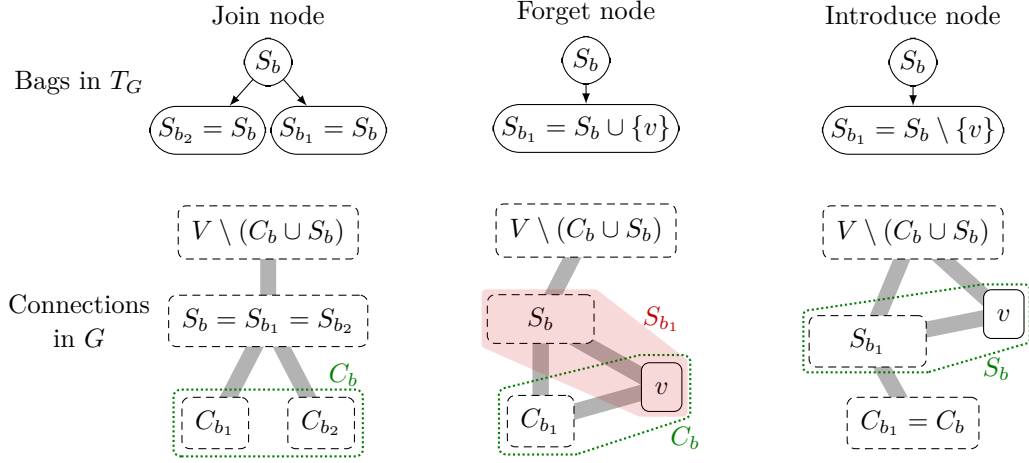
It is easy to see that Property (ii) holds as well. If we set $\alpha(v) = v$, then $v \notin \mathcal{X}$ and $\ell(v) = \infty$. Otherwise, we define $\alpha(v)$ to be either $\rho_{i_v-1}(v)$ or $\phi_{i_v-1}(v) \in T[v]$. Furthermore, any vertex w of $T[v_j]$ is anchored either to a vertex in $T[v_j] \subseteq T[v]$ or to $\rho_{\ell(w)-1}(v_j)$, since the partial anchorings fulfill Property (ii). That means w is either anchored to a vertex of $T[v]$ or, by definition of $\rho(v_j)$, to $\rho_{\ell(w)-1}(v)$.

In conclusion, $(\ell, \alpha) \in \mathcal{P}(v, \rho(v), \phi(v))$, so its cost is at least $A[v, \rho(v), \phi(v)]$.

By mathematical induction, this proves that $A[v, \rho(v), \phi(v)] = \bar{A}[v, \rho(v), \phi(v)]$, for all $v \neq r$, $\rho(v)$ and $\phi(v)$. Therefore, the cells $A[v, \rho(v), \phi(v)]$ can be computed in time $n^{O(k)}$. Define $A[r]$, with v_1, v_2, \dots, v_p being the children of r and $\rho_\emptyset := \{r, v_\emptyset, \dots, v_\emptyset\}$, as

$$A[r] = \sum_{j=1}^p \min_{\phi_i(v_j) \in T[v_j], \forall i \in [k-1]} A[v_j, \rho_\emptyset, \phi(v_j)].$$

Indeed, $A[v_j, \rho_\emptyset, \phi(v_j)]$ represents the minimum cost of a k -hop Steiner tree over $T[v_j] \cup \{r\}$ that is rooted at r and respects $\lambda_i(v_j) = \phi(v_j)$. Restricting to ρ_\emptyset prevents nodes from being anchored to other subtrees, but this is more expensive than anchoring directly to the root. Thus, $A[r]$ gives the cost of a k -hop MST. The complexity to compute $A[r]$ is linear in the size of the table, i.e. $n^{O(k)}$.



■ **Figure 5** Types of bag nodes in a *nice* tree decomposition and possible edges in G .

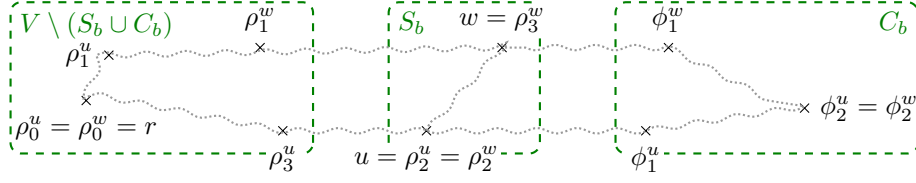
5 Metrics of Bounded Treewidth

In this section, we extend the dynamic program from Section 4 to metrics of bounded treewidth. A graph $G = (V, E)$ is said to have *treewidth* ω , if there exists a tree $T_G = (B, E_B)$ whose nodes $b \in B$ are identified with subsets $S_b \subseteq V$, called *bags*, satisfying: (i) for each edge in E , there is a bag containing both endpoints, (ii) for each vertex in V , the bags containing it form a connected subtree of T_G , and (iii) each bag contains at most $\omega + 1$ vertices. The tree T_G is called a *tree decomposition* of G . It is a *nice tree decomposition* [14] if w.r.t. a designated root b_r , every node b has one of the following four types, see Figure 5.

- *Leaf*: Its bag is empty, that is, $S_b = \emptyset$.
- *Join node*: It has two children b_1 and b_2 with $S_b = S_{b_1} = S_{b_2}$.
- *Forget node of v* : It has one child b_1 with $S_{b_1} = S_b \cup \{v\}$ and $v \notin S_b$.
- *Introduce node of v* : It has one child b_1 with $S_{b_1} = S_b \setminus \{v\}$ and $v \in S_b$.

By (ii), a vertex in V may have several introduce nodes but at most one forget node. Let C_b be the union of the bags $S_{b'}$ for all descendants b' of b , excluding vertices in S_b . Property (ii) implies that there is no edge between C_b and $V \setminus (S_b \cup C_b)$, see Figure 5, and that, for a join node, $C_{b_1} \cap C_{b_2} = \emptyset$. Given a graph of treewidth ω , we can compute a nice tree decomposition with $|B| = O(n\omega)$ in polynomial time [14]. W.l.o.g. our input is a nice tree decomposition T_G .

The dynamic program. Choose a root node b_r whose bag contains the root r of the k -hop MST which we aim to compute. To extend the dynamic programming approach from Section 4 to nice tree decompositions, we again compute cells in a bottom-up fashion, now in T_G . A key difference lies in the fact that, here, a node b in T_G corresponds to several vertices in G , so we require anchoring guarantees for every vertex in S_b . A DP cell, indexed by a bag b and $O(n^{\omega k})$ anchoring guarantees, computes a minimum cost LAP on C_b that respects these guarantees. Thankfully, the structure of the nice tree decomposition enables us to recurse in an organized manner and construct the cells consistently. Join nodes combine previous results. Forget nodes decide the label and anchoring of the corresponding vertex and possibly new anchoring guarantees needed due to forgetting it. Introduce nodes deduce anchoring guarantees about the introduced vertex from previous knowledge.



■ **Figure 6** Possible values of ρ_i and ϕ_i for two vertices u and w . Note that $\phi_3^u = \phi_3^w = v_\emptyset$.

Fix a bag $b \in B$. Its *anchoring guarantees* are given by $\phi(b) = \{\phi_i^u(b) \mid i \in [k-1] \wedge u \in S_b\}$ and $\rho(b) = \{\rho_i^u(b) \mid i \in [k-1] \wedge u \in S_b\}$, with all $\phi_i^u(b) \in \{v_\emptyset\} \cup C_b$ and $\rho_i^u(b) \in \{v_\emptyset\} \cup V \setminus C_b$. We additionally set $\phi_0^u(b) = v_\emptyset$ and $\rho_0^u(b) = r$ for all u , see Figure 6. We use these anchoring guarantees to define a subset of partial LAPs on C_b . Specifically, let $\mathcal{P}(b, \rho(b), \phi(b))$ be the (possibly empty) set of partial LAPs on C_b respecting the anchoring guarantees. That is, its elements (ℓ, α) satisfy:

- (i') $\phi_i^u(b)$ is the closest vertex to u in C_b of label i (or v_\emptyset if no such vertex exists);
- (ii') Each vertex u of C_b with $\ell(u) \neq \infty$ is anchored either to a vertex of C_b of label $\ell(u) - 1$ or to $\rho_{\ell(u)-1}^w(b)$ for some $w \in S_b$.
- (iii') For all i and $u, w \in S_b$, we have $d(u, \rho_i^u(b)) \leq d(u, \rho_i^w(b))$.

Intuitively, $\mathcal{P}(b, \rho(b), \phi(b))$ represents all relevant ways to extend a partial LAP on $V \setminus C_b$ to V : vertices of C_b are anchored either to a vertex of C_b or to some $\rho_i^u(b)$. Let $A[b, \rho(b), \phi(b)]$ be the minimum cost of a partial LAP on C_b in $\mathcal{P}(b, \rho(b), \phi(b))$, or ∞ if this set is empty. Note that if Property (iii') is satisfied and b is a leaf, then $A[b, \rho(b), \phi(b)] = 0$.

In the following, we define in a recursive procedure how to fill the cells $\bar{A}[b, \rho(b), \phi(b)]$ of our dynamic programming table \bar{A} for each node $b \in B$. The goal will be to again show that $\bar{A} = A$. First, as an easy special case, if Property (iii') is not respected by $\rho(b)$, we set \bar{A} to infinity, which matches with the fact that $\mathcal{P}(b, \rho(b), \phi(b))$ is empty and $A = \infty$. Next, we describe how to compute \bar{A} depending on the type of the node b when (iii') is respected.

Leaves: Node b has no child and $S_b = \emptyset$. We set $\bar{A}[b, \rho(b), \phi(b)] = 0$.

Join nodes: Node b has children b_1, b_2 with $S_{b_1} = S_{b_2} = S_b$ and $C_{b_1} \cup C_{b_2} = C_b$ and $C_{b_1} \cap C_{b_2} = \emptyset$. Intuitively, the objective is to independently query partial solutions on each C_{b_j} . We compute sets of possible values for $\rho_i^u(b_j)$ and $\phi_i^u(b_j)$ which define sets of partial LAPs on each C_{b_j} respecting such guarantees. These possible values are determined such that the minimum cost of a combination of any two partial LAPs on C_{b_1} and C_{b_2} in these sets equals $A[b, \rho(b), \phi(b)]$. Here, the $\rho_i^u(b_j)$ need to be equal to the closest anchoring guarantee outside of C_{b_j} . The $\phi_i^u(b_j)$ may take any value not contradicting $\phi(b)$. Specifically, for both $j \in \{1, 2\}$, $i \in [k-1]$ and $u \in S_b$:

- We set $\rho_i^u(b_j) = \text{closest}_u \{ \{ \rho_i^u(b) \} \cup \{ \phi_i^w(b) \mid w \in S_b \wedge \phi_i^w(b) \notin C_{b_j} \} \}$.
- If $\phi_i^u(b) \in C_{b_j}$, then we set $\Phi_i^u(b_j) = \{ \phi_i^u(b) \}$. Otherwise, we set

$$\Phi_i^u(b_j) = \{ x \in C_{b_j} \cup \{v_\emptyset\} \mid \text{for all } z \in S_b, \text{ we have } d(z, \phi_i^z(b)) \leq d(z, x) \quad (\star) \}.$$

where (\star) ensures that $\phi_i^z(b)$ is the vertex in C_b that is closest to z .

We then define

$$\bar{A}[b, \rho(b), \phi(b)] = \sum_{j \in \{1, 2\}} \min_{\phi_i^u(b_j) \in \Phi_i^u(b_j), \forall i \in [k-1], u \in S_{b_j}} \bar{A}[b_j, \rho(b_j), \phi(b_j)]. \quad (3)$$

Forget node of v : We have $S_{b_1} = S_b \cup \{v\}$ and $C_{b_1} = C_b \setminus \{v\}$. There is no edge between v and $V \setminus (C_b \cup S_b)$. In this node, we want to define the label i_v of v and the corresponding anchoring of v . However, $\phi(b)$ may not contain sufficient information for deciding i_v , since v can be far away from S_b . We therefore need to consider all possible values for i_v , that is all values that are consistent with the guarantees $\phi(b)$. We first define the set \mathbf{I}_v of possible labels of v that do not contradict $\phi(b)$, then proceed to define possible values of $\phi(b_1)$, $\rho(b_1)$, and finally, we express the cost to anchor v .

Let \mathbf{I}_v be the set of labels i such that for all $u \in S_b$, we have $d(u, v) \geq d(u, \phi_i^u(b))$ and for all $i' \neq i$, we have $\phi_{i'}^u(b) \neq v$. In other words, if there is a label i and $u \in S_b$ with $\phi_i^u(b) = v$, then \mathbf{I}_v cannot contain any other label: In order to respect the guarantee $\phi_i^u(b) = v$, we must have $i_v = i$. Moreover, if there exists some $u \in S_b$ and i such that $\phi_i^u(b)$ is further from u than v , then \mathbf{I}_v cannot contain i as it would contradict the definition of $\phi_i^u(b)$. If $v \notin \mathcal{X}$ and no $\phi_i^u(b)$ equals v , we include ∞ in \mathbf{I}_v as v does not need to have a finite label in order to respect the guarantees $\phi_i(b)$. If \mathbf{I}_v is empty, set $\bar{A}[b, \rho(b), \phi(b)]$ to be infinite since it is impossible to label v while respecting the guarantees $\phi_i(b)$. Assume now that \mathbf{I}_v is not empty.

The values $\phi_i^u(b_1)$ can take any value in C_{b_1} not contradicting $\phi(b)$. Specifically, for $u \in S_b$, if $\phi_i^u(b) \neq v$ let $\Phi_i^u(b_1) = \{\phi_i^u(b)\}$, and if $\phi_i^u(b) = v$, let

$$\Phi_i^u(b_1) = \{x \in C_{b_1} \cup \{v_\emptyset\} \mid d(u, v) \leq d(u, x)\}.$$

Indeed, if $\phi_i^u(b) = v$, then we need to provide a new guarantee for $\phi_i^u(b_1)$, as $v \in S_{b_1}$, which must be further from u than v . We also define

$$\Phi_i^v(b_1) = \{x \in C_{b_1} \cup \{v_\emptyset\} \mid \text{for all } u \in S_b, \text{ we have } d(u, \phi_i^u(b)) \leq d(u, x) \quad (\star)\}.$$

Again, (\star) must be satisfied since $\phi_i^u(b)$ is the vertex in C_b which is closest to u .

For the remainder, fix some $i_v \in \mathbf{I}_v$. In the case where $i_v = \infty$, we need not consider ρ_{i_v} 's. Otherwise, any path from v to a vertex in $V \setminus C_b$ passes through S_b . Therefore, $\rho_{i_v}^v(b_1)$ is determined by $\rho_{i_v}^v(b_1) = \text{closest}_v\{\rho_i^u(b) \mid u \in S_b\}$ for $i \neq i_v$, and $\rho_{i_v}^v(b_1) = v$. Similarly, for $u \in S_b$, let $\rho_{i_v}^u(b_1) = \rho_i^u(b)$ for $i \neq i_v$, and $\rho_{i_v}^u(b_1) = \text{closest}_u\{\rho_{i_v}^u(b), v\}$.

Additionally, we charge a cost of c_{i_v} for anchoring v . If $i_v = \infty$ then set $c_{i_v} := 0$. Otherwise, set $c_{i_v} := d(v, \text{closest}_v\{\phi_{i_v-1}^v(b_1), \rho_{i_v-1}^v(b_1)\})$.

We then define, with $\rho(b_1)$ depending on i_v and c_{i_v} depending on $\phi_{i_v}^u(b_1)$,

$$\bar{A}[b, \rho(b), \phi(b)] = \min_{i_v \in \mathbf{I}_v} \min_{\phi_i^u(b_1) \in \Phi_i^u(b_1), \forall i \in [k-1], u \in S_{b_j}} \left(c_{i_v} + \bar{A}[b_1, \rho(b_1), \phi(b_1)] \right). \quad (4)$$

Introduce node of v : In this case, b has one child b_1 with $S_{b_1} = S_b \setminus \{v\}$ and $C_{b_1} = C_b$. There is no edge between v and $C_b = C_{b_1}$ as $v \notin S_{b_1} \cup C_{b_1}$, see Figure 5. If there is an i with $\phi_i^v(b) \neq \text{closest}_v\{\phi_i^u(b) \mid u \in S_{b_1}\}$ then $\bar{A}[b, \rho(b), \phi(b)]$ is infinite since the shortest v - $\phi_i^v(b)$ path has to pass through a vertex of S_b by the above observation. Otherwise, the guarantees do not change, so we define $\rho(b_1) = \rho(b)$, $\phi(b_1) = \phi(b)$, and we set

$$\bar{A}[b, \rho(b), \phi(b)] = \bar{A}[b_1, \rho(b_1), \phi(b_1)]. \quad (5)$$

One can check that the running time to compute \bar{A} is $n^{O(\omega k)}$. The correctness of this dynamic program, i.e., the proof of the equality $A = \bar{A}$ and the specification of the final queries on the root bag cells are discussed in the full version of the paper.

6 The k -hop MŠT Problem in Metrics of Bounded Highway Dimension

In this section, we consider k -hop MŠT in metrics of bounded highway dimension and give the proof of Theorem 3. Denoting $B_r(v) = \{u \in V \mid d(u, v) \leq r\}$, the highway dimension of a graph is defined as follows in [17]. Given a universal constant $c > 4$, the *highway dimension* of a graph G is the smallest integer h such that for every $r \geq 0$ and $v \in V$, there is a set of h vertices in $B_{cr}(v)$ that hits all shortest paths of length more than r that lie entirely in $B_{cr}(v)$. Before stating the results, we first define a δ -net of a graph, which is informally a subset of vertices which are far from each other, while every vertex in the graph is close to this subset. Formally, a δ -net of a graph G , is a subset U of V such that for all $u \in V$, there exists $v \in U$ with $d(u, v) \leq \delta$ and for all $u, v \in U$, we have $d(u, v) > \delta$.

Feldmann et al. [17] proved the following result, which gives sufficient conditions for a problem to admit a *QPTAS* on graphs of constant highway dimension.

► **Theorem 7** (Reformulation of [17, Theorem 8.1]). *For a graph G of constant highway dimension and a problem \mathcal{P} satisfying conditions 1-6 below, a $(1 + \varepsilon)$ -approximation can be computed in quasi-polynomial time.*

1. An optimum solution of \mathcal{P} can be computed in time $n^{O(\omega)}$ for a graph of treewidth ω ;
2. A constant-approximation of \mathcal{P} on metric graphs can be computed in polynomial time;
3. The diameter of the graph can be assumed to be $O(n \cdot \text{OPT}_G)$, where OPT_G is the cost of an optimal solution in G ;
4. An optimum solution for \mathcal{P} on a δ -net U has cost at most $\text{OPT}_G + O(n\delta)$;
5. The objective function of \mathcal{P} is linear in the edge cost;
6. A solution for \mathcal{P} on a δ -net U can be converted to a solution on V for a cost of $O(n\delta)$.

We now show that our main result, Theorem 2, together with a previously known result, leads to Theorem 3, by using a slight variation of Theorem 7 to allow resource augmentation.

Proof of Theorem 3. Applying Theorem 7, it remains to verify that the k -hop MŠT problem satisfies its six conditions, for k constant, if we allow the algorithm to use one more hop (i.e., computing a $(k + 1)$ -hop Steiner tree) than the optimal solution of cost OPT_k to which we compare it. The conditions and the explanation of why they are fulfilled are detailed below.

1. Theorem 2 states precisely this condition for k -hop MŠT.
2. For k -hop MŠT in metric graphs, there exists a $(1.52 \cdot 9^{k-2})$ -approximative algorithm [25].
3. The diameter of G can be assumed to be $O(n \cdot \text{OPT}_k)$ since edges of cost larger than $1.52 \cdot 9^{k-2} \cdot \text{OPT}_k$ can be deleted after computing the approximation of OPT_k from [25].
4. Consider an optimum k -hop MŠT on V and move each vertex not in U to the closest vertex in U . This induces an extra cost of $O(n\delta)$ and is a solution on U .
5. The objective function of k -hop MŠT is indeed linear in the edge cost.
6. This condition requires an additional hop. We claim that a solution for k -hop MŠT on a δ -net U can be converted to a $(k + 1)$ -hop Steiner tree on V for an additional cost of $O(n\delta)$. Indeed, given a k -hop Steiner tree on U , we can anchor all vertices from $V \setminus U$ to their closest vertex in U for an additional cost of $O(n\delta)$ and obtain a $(k + 1)$ -hop Steiner tree. This procedure of extending a solution is performed exactly once in the underlying algorithm. Therefore we can allow the algorithm to use one more hop on G than the solution on U . Note that this property is not stated explicitly in [17]. ◀

References

- 1 Ittai Abraham, Daniel Delling, Amos Fiat, Andrew V. Goldberg, and Renato F. Werneck. Highway dimension and provably efficient shortest path algorithms. *J. ACM*, 63(5):41:1–41:26, 2016.
- 2 Ittai Abraham, Amos Fiat, Andrew V. Goldberg, and Renato F. Werneck. Highway dimension, shortest paths, and provably efficient algorithms. In *Proceedings of the twenty-first annual ACM-SIAM symposium on Discrete Algorithms*, pages 782–793, 2010.
- 3 Laurent Alfandari and Vangelis Th. Paschos. Approximating minimum spanning tree of depth 2. *International Transactions in Operational Research*, 6(6):607–622, 1999.
- 4 Ernst Althaus, Stefan Funke, Sarel Har-Peled, Jochen Könemann, Edgar A. Ramos, and Martin Skutella. Approximating k -hop minimum-spanning trees. *Oper. Res. Lett.*, 33(2):115–120, 2005.
- 5 Omer Angel, Abraham D. Flaxman, and David B. Wilson. A sharp threshold for minimum bounded-depth and bounded-diameter spanning trees and steiner trees in random networks. *Combinatorica*, 32(1):1–33, 2012.
- 6 Sunil Arya, Gautam Das, David M. Mount, Jeffrey S. Salowe, and Michiel Smid. Euclidean spanners: Short, thin, and lanky. In *Proceedings of the Twenty-Seventh Annual ACM Symposium on Theory of Computing*, page 489–498, 1995.
- 7 Anantaram Balakrishnan and Kemal Altinkemer. Using a hop-constrained model to generate alternative communication network design. *INFORMS Journal on Computing*, 4(2):192–205, 1992.
- 8 Judit Bar-Ilan, Guy Kortsarz, and David Peleg. Generalized submodular cover problems and applications. *Theor. Comput. Sci.*, 250(1-2):179–200, 2001.
- 9 Marshall Bern and Paul Plassmann. The steiner problem with edge lengths 1 and 2. *Information Processing Letters*, 32(4), 1989.
- 10 Hans L. Bodlaender. A linear-time algorithm for finding tree-decompositions of small treewidth. *SIAM J. Comput.*, 25(6):1305–1317, 1996.
- 11 Paz Carmi, Lilach Chaitman-Yerushalmi, and Ohad Trabelsi. Bounded-hop communication networks. *Algorithmica*, 80(11):3050–3077, 2018.
- 12 Markus Chimani, Petra Mutzel, and Bernd Zey. Improved steiner tree algorithms for bounded treewidth. *Journal of Discrete Algorithms*, 16:67–78, 2012.
- 13 Markus Chimani and Joachim Spoerhase. Network Design Problems with Bounded Distances via Shallow-Light Steiner Trees. In *32nd International Symposium on Theoretical Aspects of Computer Science (STACS 2015)*, volume 30, pages 238–248, 2015.
- 14 Marek Cygan, Fedor V. Fomin, Łukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michał Pilipczuk, and Saket Saurabh. Treewidth. In *Parameterized algorithms*, chapter 7, pages 151–244. Springer, Cham, 2015. doi:10.1007/978-3-319-21275-3_7.
- 15 Geir Dahl, Luís Gouveia, and Cristina Requejo. On formulations and methods for the hop-constrained minimum spanning tree problem. In Mauricio G. C. Resende and Panos M. Pardalos, editors, *Handbook of Optimization in Telecommunications*, pages 493–515. Springer, Boston, MA, 2006.
- 16 Michael Elkin and Shay Solomon. Narrow-shallow-low-light trees with and without steiner points. *SIAM Journal on Discrete Mathematics*, 25(1):181–210, 2011.
- 17 Andreas Emil Feldmann, Wai Shing Fung, Jochen Könemann, and Ian Post. A $(1+\epsilon)$ -embedding of low highway dimension graphs into bounded treewidth graphs. *SIAM Journal on Computing*, 47(4):1667–1704, 2018.
- 18 Luis Gouveia. Using the miller-tucker-zemlin constraints to formulate a minimal spanning tree problem with hop constraints. *Comput. Oper. Res.*, 22(9):959–970, 1995.
- 19 Luis Gouveia, Luidi Simonetti, and Eduardo Uchoa. Modeling hop-constrained and diameter-constrained minimum spanning tree problems as steiner tree problems over layered graphs. *Math. Program.*, 128(1-2):123–148, 2011.

- 20 Sudipto Guha and Samir Khuller. Greedy strikes back: Improved facility location algorithms. *Journal of Algorithms*, 31(1):228–248, 1999.
- 21 Anupam Gupta, Robert Krauthgamer, and James R. Lee. Bounded geometries, fractals, and low-distortion embeddings. In *FOCS*, pages 534–543. IEEE Computer Society, 2003.
- 22 Martin Haenggi. Twelve reasons not to route over many short hops. *VTC2004-Fall*, 5:3130–3134 Vol. 5, 2004.
- 23 MohammadTaghi Hajiaghayi, Guy Kortsarz, and Mohammad R. Salavatipour. Approximating buy-at-bulk and shallow-light k -steiner trees. *Algorithmica*, 53(1):89–103, 2009.
- 24 Vojtěch Jarník and Miloš Kössler. O minimálních grafech, obsahujících n daných bodů. *Časopis pro pěstování matematiky a fyziky*, 63(8):223–235, 1934.
- 25 Erez Kantor and David Peleg. Approximate hierarchical facility location and applications to the bounded depth steiner tree and range assignment problems. *J. Discrete Algorithms*, 7(3):341–362, 2009.
- 26 Bernhard Korte and Jaroslav Nešetřil. Vojtěch Jarník’s work in combinatorial optimization. *Discrete Mathematics*, 235(1-3):1–17, 2001.
- 27 Guy Kortsarz and David Peleg. Approximating shallow-light trees. In *Proceedings of the Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, page 103–110, 1997.
- 28 Sören Laue and Domagoj Matijević. Approximating k -hop minimum spanning trees in euclidean metrics. *Inf. Process. Lett.*, 107(3-4):96–101, 2008.
- 29 Shi Li. A 1.488 approximation algorithm for the uncapacitated facility location problem. *Information and Computation*, 222:45–58, 2013.
- 30 Prabhu Manyem and Matthias FM Stallmann. Some approximation results in multicasting. Technical report, North Carolina State University at Raleigh, USA, 1996.
- 31 Vikram Raj Saksena. Topological analysis of packet networks. *IEEE Journal on Selected Areas in Communications*, 7(8):1243–1252, 1989.
- 32 Stefan Voß. The steiner tree problem with hop constraints. *Annals OR*, 86:321–345, 1999.