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

Martin Böhm[1], Ruben Hoeksma[2], Nicole Megow[1], Lukas Nölke[1], and Bertrand Simon[1]

[1] University of Bremen, Germany
{martin.boehm,nicole.megow,noelke,bsimon}@uni-bremen.de
[2] University of Twente, The Netherlands
r.p.hoeksma@utwente.nl

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.

Keywords: k-hop Steiner tree · dynamic programming · network design

1 Introduction

We are given a finite metric space (V, d) with $|V| = n$ points and distance function $\mathrm{d} : V \times V \to \mathbb{Q}_+$, a set of terminals $\mathcal{X} \subseteq V$ and a root $r \in \mathcal{X}$, as well as an integer $k \geq 1$. A *k-hop Steiner tree* is a tree $T = (V_T, E_T)$ rooted at r that spans all points in $\mathcal{X}$ and has a depth of at most k. That is, $\mathcal{X} \subseteq V_T \subseteq V$ and for $v \in V_T$, the number of edges in the r-v path in T is at most k. The cost of a Steiner tree is the sum of edge costs $\sum_{\{u,v\} \in E_T} \mathrm{d}(u, v)$, with edge costs given by d. We consider the *minimum-cost k-hop Steiner tree problem* (k-hop MŠT problem[3]) 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 by limiting the amplifying effect of link failures. There exists a multitude of applications; see, e.g., [6, 9, 13, 16, 17, 19, 27, 28].

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 $\mathrm{d}(u, v)$. 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,

[3] For brevity and as homage to the work of Jarník and Kössler [22], we use the Czech letter Š to distinguish Steiner trees from spanning trees in MŠT resp. MST.

a tree metric, or a metric of constant treewidth ω. 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 (UFL) 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 [3], even for graph metrics, while the Minimum Steiner Tree problem is polynomial-time solvable on graphs of bounded treewidth [10].

When considering metrics more general than those of bounded treewidth, several hardness results are known. Bern and Plassmann [8] 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 [2]. Thus, these problems do not admit a PTAS, unless P = NP. Manyem and Stallmann [26] show that k-hop MŠT on a graph with unit-weight edges and 2-hop MST cannot admit a constant 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 [2] 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 [23] 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. [3] 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 (HST) and then solves this problem optimally. For constant k, Laue and Matijević [24] 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 Kantor and Peleg [21], the approximation factor $1.52 \cdot 9^{k-2}$ is prohibitively high. For $k = 2$, a nearly optimal algorithm is known. Since the best known approximation ratio for metric UFL is 1.488 [25] and the best known lower bound is 1.463 [18], these bounds are valid for metric 2-hop MST as well.

The bounded-diameter minimum Steiner tree problem [17, 21] 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 [21]. For non-metric cost functions, an $o(\log n)$-approximation algorithm has been ruled out, assuming P$\neq$NP [7].

Further, shallow-light and buy-at-bulk Steiner trees [5, 11, 14, 20, 23] 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. the metric distance instead of the number of edges on the path. Elkin and Solomon [14] additionally bound the number of hops, but do so by $O(\log n)$ to bound the other measures of interest. Chimani and Spoerhase [11] 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 [4].

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 [3] which is tailored to HSTs. While the DP for planar graphs in [24] 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 [1] 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. Building on a framework from [15, Theorem 8.1], we obtain the following result, which is proved in Section 6.

Theorem 3. *For a metric induced by a graph of bounded highway dimension and 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 [3].

2 Preliminaries

Let (V, d) be a metric induced by the graph $G = (V, E)$ with terminals $\mathcal{X} \subseteq V$ and $r \in \mathcal{X}$. In order to break ties consistently, we assume shortest paths in G to be unique. This can be archived by adding some sufficiently small noise to the input slightly moving each point. A k-hop MŠT after this transformation, is also optimal for the original instance. Additionally, we assume that G is the (by the previous assumption unique) *minimal graph inducing* (V, d). That is, there is no edge in G that can be removed without changing some shortest path.

When working in a metric induced by a graph G, it is useful to view a k-hop Steiner tree T with root r as two assignments on the vertices of G. Specifically, for $U \subseteq V$, call a map $\ell : U \longrightarrow \{0, 1, \ldots, k\} \cup \{\infty\}$ a *labeling on* U and $\alpha : U \setminus \{r\} \longrightarrow V$ an *anchoring on* U. The label of u indicates its depth in T, i.e., the number of edges on the u-r path, while its anchor describes the first vertex after u on this path, its parent in T. The label 0 is only assigned to the root r. Together, a labeling and anchoring represent a partial k-hop Steiner tree.

Definition 1. *A pair (ℓ, α) is called* 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 $\alpha^{-1}(u) = \{u\}$.*

A LAP (ℓ, α) on V characterizes a k-hop Steiner tree T. The cost of T is equal to the *cost of the anchoring* α given by $\sum_{u \in U \setminus \{r\}} \mathrm{d}(u, \alpha(u))$. We also call this the *cost of the LAP* (ℓ, α). We say $\mathrm{d}(u, \alpha(u))$ is the *cost to anchor* u, which is zero if $u \notin T$. The fact that $u \notin T$ is encoded by $\ell(u) = \infty$ and $\alpha(u) = u$ in a LAP. It is not hard to see that for a given LAP (ℓ, α) on V of minimum cost, the labeling ℓ can be computed from the anchoring α in polynomial time and vice-versa. When $U \neq V$, we may say *partial* LAP to emphasize that the LAP only represents a portion of T, namely the edges between U and its anchors.

To avoid confusion and to differentiate between the (partial) k-hop Steiner tree T and the metric space (especially when induced by another tree), we use the above LAP representation of T. Specifically, when talking about distances or closeness, we refer to distances in G. Given a point v and a set $S \subseteq V$, denote by $\mathrm{closest}_v(S)$ the (unique) element of S with minimum distance to v.

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 $[k] = \{1, 2, \ldots, k\}$. 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 $V = \{v_1, v_2, \ldots, v_n\}$ 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 $\mathrm{OPT} = (\ell, \alpha)$ 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 1. 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.

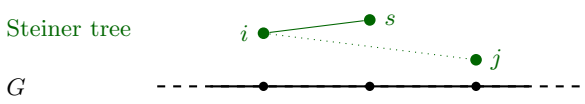

Fig. 1. The optimal k-hop MŠT never attaches j to i if $\ell(i) > \ell(s)$.

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 T 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 $[a, b] = \emptyset$.

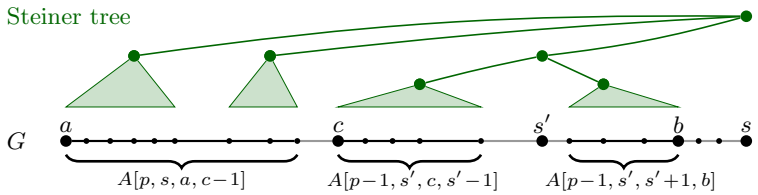

Fig. 2. Computation of $A[p, s, a, b]$ with three recursive calls.

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 $A[p, s, a, a] = \mathrm{d}(v_s, v_a)$.
3. If $p = 1$, then $A[1, s, a, b] = \sum_{x \in [a,b]} \mathrm{d}(v_s, v_x)$ (all vertices anchored to v_s).
4. If $p > 1$, consider the right-most child $v_{s'}$ of v_s in T such that $s' \in [a, b]$. The sub-tree of T 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

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

See Figure 2 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 (Theorem 1). Due to the key observation above, $A[p, s, a, b]$ correctly computes the minimum cost of a p-hop spanning tree T 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 4. *In tree metrics, k-hop MŠT can be solved exactly in time $n^{O(k)}$.*

In this section, we construct a dynamic program for k-hop MŠT on tree metrics. Consider a tree metric induced by a tree $T = (V, E)$ with root r. For $v \in V$, denote by $T[v]$ the set of vertices in the subtree of T rooted at v.

We start by giving a high-level overview of our approach for computing the minimum cost k-hop Steiner tree 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 $\ell(\phi_i) = \ell(\rho_i) = i$

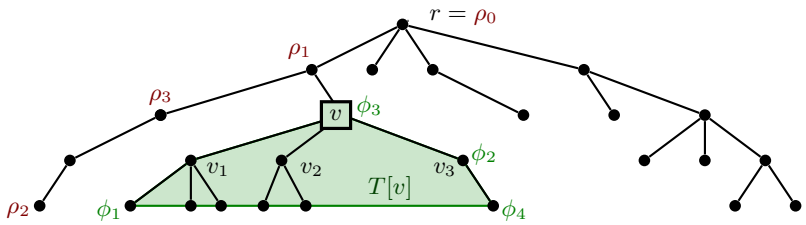

Fig. 3. Possible values for $\rho(v)$ and $\phi(v)$ in the tree T for $k = 5$. Note that $\rho_4 = v_\emptyset$.

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 1) 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 $\phi(v) = \big(\phi_1(v), \dots, \phi_{k-1}(v)\big)$ and $\rho(v) = \big(\rho_1(v), \dots, \rho_{k-1}(v)\big)$ 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 3. We may use ϕ_i or ρ_i when referring to the fixed vertex v.

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:

(A) For all i, we have $\phi_i = \lambda_i(v)$. In particular, if $\phi_i = \phi_j$ and $i \neq j$, then $\phi_i = v_\emptyset$.
(B) 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 $\rho_{\ell(w)-1}$. 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. This happens by anchoring vertices of $T[v]$ either to another vertex in $T[v]$ or to some ρ_i. Therefore, if ρ_i is used, it should be the closest vertex to v outside of $T[v]$ for which $\ell'(\rho_i) = i$. 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 $\phi_i = \phi_i(v)$ (which may be in the subtree of a different child), or to a $\rho_i = \rho_i(v)$. The anchoring guarantees ϕ_i are necessary to determine the anchoring guarantees $\rho_i(v_j)$ for the children of v.

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. The heart of our computation of $A[v, \rho(v), \phi(v)]$ is the recursive formula

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

Here, c_v is the cost of anchoring v while $\boldsymbol{\Phi}_i(v_j)$ and $\boldsymbol{\rho}(v_j)$ 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 $\boldsymbol{\Phi}_i(v_j)$, $\boldsymbol{\rho}(v_j)$ and c_v follow.

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

$\mathrm{d}(v,w) < \mathrm{d}(v,\phi_i)$ would contradict the choice of ϕ_i as the vertex in $T[v]$ of label i closest to v.

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

We now define c_v. If $v \notin \mathcal{X}$ and no ϕ_i equals v, then $c_v \coloneqq 0$ as v cannot have a label respecting ((A)). Next, if there exists a unique i_v such that $\phi_{i_v} = v$, let $c_v \coloneqq \mathrm{d}(v, \mathrm{closest}_v(\rho_{i_v-1}, \phi_{i_v-1}))$. In all other cases, set $c_v \coloneqq \infty$ as the values of $\phi(v)$ are contradictory. Recall, that $\mathrm{closest}_v(\rho_{i_v-1}, \phi_{i_v-1})$ denotes the vertex closer to v, which is the one v will be anchored to in a solution of minimum cost.

Proof of Theorem 4. We prove that $\bar{A}[v, \rho(v), \phi(v)]$, as defined in Equation (1), is equal to $A[v, \rho(v), \phi(v)]$, for $v \neq r$ and every $\rho(v)$, $\phi(v)$. In the case where $c_v = \infty$, this implies $\mathcal{P}(v, \rho(v), \phi(v)) = \emptyset$, so both $A[v, \rho(v), \phi(v)]$ and $\bar{A}[v, \rho(v), \phi(v)]$ are infinite. From now on, assume that c_v is finite.

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 (A) and (B) 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 $\phi_i(v_j) \in \boldsymbol{\Phi}_i(v_j)$. Also, set $\rho(v_j) = \boldsymbol{\rho}(v_j)$ 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 (A) follows directly from the choice of $\phi_i(v_j) = \lambda_i(v_j)$.

For Property (B), 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 $\ell_w \coloneqq \ell(w)-1$. Note that by definition of $\boldsymbol{\rho}(v_j)$, we have that $\rho_{\ell_w}(v_j)$ is equal to ρ_{ℓ_w} or ϕ_{ℓ_w}, so $\ell(\rho_{\ell_w}(v_j)) = \ell_w$. As α is an anchoring of minimal cost (w.r.t. the given guarantees), w is anchored to $\alpha(w) = \mathrm{closest}_w\{x \in T[v] \cup \{\rho_{\ell_w}\} \mid \ell(x) = \ell_w\}$, so $\alpha(w) = \mathrm{closest}_{v_j}(\rho_{\ell_w}, \phi_{\ell_w})$. If $\phi_{\ell_w} \in T[v_j]$, then $\boldsymbol{\rho}_{\ell_w}(v_j) = \rho_{\ell_w} = \alpha(w)$ since w is not anchored to a vertex in $T[v_j]$. If $\phi_{\ell_w} \notin T[v_j]$, then by definition of $\boldsymbol{\rho}(v_j)$, we have $\boldsymbol{\rho}_{\ell_w}(v_j) = \mathrm{closest}_{v_j}(\rho_{\ell_w}, \phi_{\ell_w}) = \alpha(w)$.

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 $\alpha(v) = \mathrm{closest}_v(\rho_{i_v-1}, \phi_{i_v-1})$ 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 $\phi_{i_v} = v$. We then let $\ell(v) = i_v$ and anchor v to $\mathrm{closest}_v(\rho_{i_v-1}, \phi_{i_v-1})$. We show that this yields an element of $\mathcal{P}(v, \rho(v), \phi(v))$.

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

It is easy to see that Property (B) holds as well. If we set $\alpha(v) = v$, then $v \notin \mathcal{X}$ and $\ell(v) = \infty$. Otherwise, we defined $\alpha(v)$ to be either ρ_{i_v-1} or $\phi_{i_v-1} \in T[v]$. 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

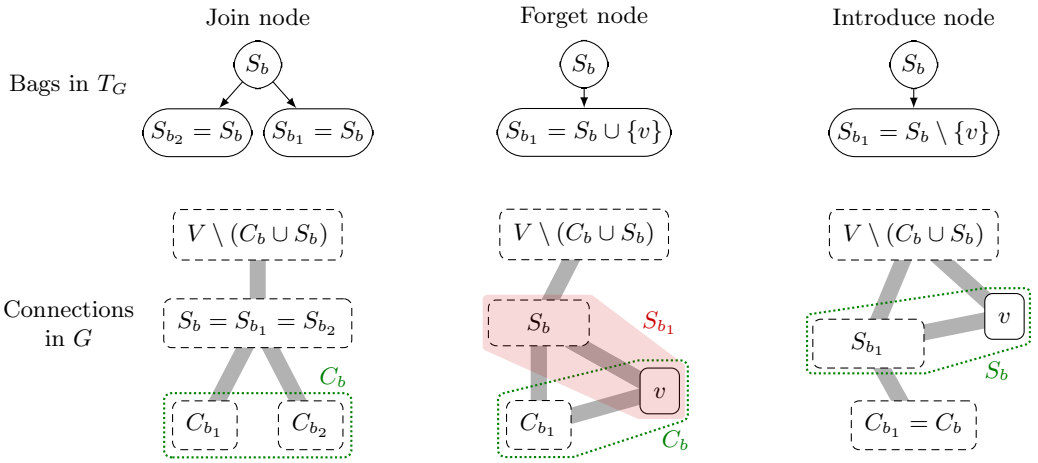

Fig. 4. Types of bag nodes in a *nice* tree decomposition and possible edges in G.

partial anchorings fulfill Property (B). That means w is either anchored to a vertex of $T[v]$ or, by definition of $\boldsymbol{\rho}(v_j)$, to $\rho_{\ell(w)-1}$.

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

We have shown that $A[v, \rho(v), \phi(v)] = \bar{A}[v, \rho(v), \phi(v)]$, for all ρ, ϕ and $v \neq r$. Therefore, the cells $A[v, \rho(v), \phi(v)]$ can be computed in time $n^{O(k)}$. Define $A[r]$, with $v_1, v_2, \ldots, v_p$ being the children of r and $\rho_\emptyset \coloneqq \{r, v_\emptyset, \ldots, v_\emptyset\}$, 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 $\rho_\emptyset$ 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 MŠT. The complexity to compute $A[r]$ is linear in the size of the table, i.e. $n^{O(k)}$. □

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* [12] if w.r.t. a designated root b_r, every node b has one of the following four types, see Figure 4.

- *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)

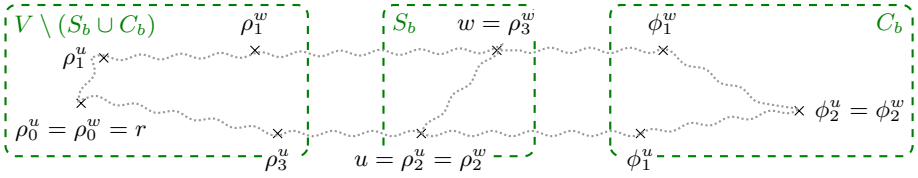

Fig. 5. Possible values of ρ_i and ϕ_i for two vertices u and w. Note that $\phi_3^u = \phi_3^w = v_\emptyset$.

implies that there is no edge between C_b and $V \setminus (S_b \cup C_b)$, see Figure 4, 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 [12]. 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 MŠT 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.

Formally, cells of the dynamic programming table A are indexed by a bag b as well as anchoring guarantees $\phi(b) = \{\phi_i^u(b) \in C_b \cup \{v_\emptyset\} \mid i \in [k-1] \wedge u \in S_b\}$ and $\rho(b) = \{\rho_i^u(b) \notin C_b \mid i \in [k-1] \wedge u \in S_b\}$. The value of the cell represents the minimum cost of anchoring C_b while respecting the given anchoring guarantees. That is $A[b, \rho(b), \phi(b)]$ equals the minimum cost of any LAP on C_b that satisfies:

(A′) $\phi_i^u(b)$ is the closest vertex to u in C_b of label i (or $v_\emptyset$ if no such vertex exists);
(B′) 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 some $\rho_{\ell(u)-1}^w(b)$.
(C′) For all i and $u, w \in S_b$, we have $\mathrm{d}(u, \rho_i^u(b)) \leq \mathrm{d}(u, \rho_i^w(b))$.

Denote by $\mathcal{P}(b, \rho(b), \phi(b))$ the set of all LAPs satisfying the above properties (which may be empty), and set $\phi_0^u(b) = v_\emptyset$ and $\rho_0^u(b) = r$ for all u and b, see Figure 5. If $\mathcal{P}(b, \rho(b), \phi(b)) = \emptyset$, then we set $A[b, \rho(b), \phi(b)] = \infty$.

For each node b, we define $\bar{A}[b, \rho(b), \phi(b)]$ as a function of cells of A corresponding to the bags of the children of b. The goal is to again show that $\bar{A} = A$. If Property (C′) is not respected by $\rho(b)$, set $\bar{A}$ to be infinite. We describe how to compute $\bar{A}$ depending on the type of the node b when Property (C′) 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 $C_{b_1} \cap C_{b_2} = \emptyset$. The objective is to independently query partial solutions on each C_{b_j}, thereby determining sets of possible values for $\rho_i^u(b_j)$ and $\phi_i^u(b_j)$ such that the minimum cost of a combination of LAPs in $\mathcal{P}(b_1, \rho(b_1), \phi(b_1))$ and $\mathcal{P}(b_2, \rho(b_2), \phi(b_2))$ 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 $\boldsymbol{\rho}_i^u(b_j) = \text{closest}_u\{\{\rho_i^u(b)\} \cup \{\phi_i^w(b) \mid w \in S_b \ \wedge \ \phi_i^w(b) \notin C_{b_j}\}\}$.
– If $\phi_i^u(b) \in C_{b_j}$, then we set $\boldsymbol{\Phi}_i^u(b_j) = \{\phi_i^u(b)\}$. Otherwise, we set

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

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

We then define

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

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. We first define the set $\boldsymbol{I}_v$ 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 $\boldsymbol{I}_v$ be the set containing all labels i such that for all $u \in S_b$, we have $\mathrm{d}(u, v) \geq \mathrm{d}(u, \phi_i^u(b))$ and for all $i' \neq i$, we have $\phi_{i'}^u(b) \neq v$. In other words, if there is a label i and some $u \in S_b$ with $\phi_i^u(b) = v$, then $\boldsymbol{I}_v$ cannot contain any other label. Moreover, if there exists some $u \in S_b$ and i such that $\phi_i^u(b)$ is further from u than v, then $\boldsymbol{I}_v$ 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 $\boldsymbol{I}_v$. If $\boldsymbol{I}_v$ is empty, set $\bar{A}[b, \rho(b), \phi(b)]$ to be infinite. Assume now that $\boldsymbol{I}_v$ 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 $\boldsymbol{\Phi}_i^u(b_1) = \{\phi_i^u(b)\}$, and if $\phi_i^u(b) = v$, let

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

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

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

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

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

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

$$\bar{A}[b, \rho(b), \phi(b)] = \min_{i_v \in \boldsymbol{I}_v} \min_{\phi_i^u(b_1) \in \boldsymbol{\Phi}_i^u(b_1),\ \forall i \in [k-1],\, u \in S_{b_j}} \Big(c_{i_v} + A[b_1, \boldsymbol{\rho}(b_1), \phi(b_1)]\Big).$$

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 4. 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 values of ρ and ϕ are not changed and we define $\bar{A}[b, \rho(b), \phi(b)] = A[b_1, \rho(b_1), \phi(b_1)]$.

One can check that the cost to compute $\bar{A}$ is $n^{O(\omega k)}$.

We proceed to prove that, for all bag b and values $\rho(b)$, $\phi(b)$, the quantity $\bar{A}[b, \rho(b), \phi(b)]$ matches the minimum cost $A[b, \rho(b), \phi(b)]$ of a labeling-anchoring pair in $\mathcal{P}(b, \rho(b), \phi(b))$.

First direction, $A[b,\rho(b),\phi(b)] \geq \bar{A}[b,\rho(b),\phi(b)]$. Consider a bag b and any values $\rho(b)$ and $\phi(b)$. If b is a leaf, the result holds as both sides are zero. Consider a LAP $(\ell,\alpha) \in \mathcal{P}(b,\rho(b),\phi(b))$ for which the value $A[b,\rho(b),\phi(b)]$ is attained. If no such LAP exists, then $A[b,\rho(b),\phi(b)] = \infty$ and the inequality holds. There are three different cases depending on the type of b. For each case, we focus on a bag node b_j child of b and define values $\rho(b_j)$ and $\phi(b_j)$. We show that the restriction of (ℓ,α) to C_{b_j} belongs to $\mathcal{P}(b_j,\rho(b_j),\phi(b_j))$. We then show that the value of cell $A[b_j,\rho(b_j),\phi(b_j)]$ was considered in the computation of $\bar{A}[b,\rho(b),\phi(b)]$, i.e., each $\phi_i^u(b_j)$ belongs to the corresponding $\boldsymbol{\Phi}_i^u(b_j)$ and $\rho(b_j) = \boldsymbol{\rho}(b_j)$.

We define $\phi_i^u(b_j)$ as the closest vertex to u in C_{b_j} of label i (with respect to ℓ), and $\rho_i^u(b_j) = \boldsymbol{\rho}_i^u(b_j)$, which is defined in Section 5 according to the type of bag b. This way, $\phi_i^u(b_j)$ automatically satisfies Property (A′) for (ℓ,α) restricted to C_{b_j}. In order to prove that the restriction of (ℓ,α) belongs to $\mathcal{P}(b_j,\phi(b_j),\rho(b_j)$, it therefore remains to show that this LAP also respects Properties (B′) and (C′) regarding $b_j,\phi(b_j),\rho(b_j)$.

Once all three requirements ($\phi_i^u(b_j) \in \boldsymbol{\Phi}_i^u(b_j)$, Properties (B′) and (C′)) are verified, we use the definitions of $\bar{A}[b,\rho(b),\phi(b)]$ for each bag type in Section 5 to arrive at the desired inequality $A[b,\rho(b),\phi(b)] \geq \bar{A}[b,\rho(b),\phi(b)]$.

- *Join nodes:* For a join node b with children b_1,b_2, we focus on a single child b_j. We first show that each $\phi_i^u(b_j)$ belong to $\boldsymbol{\Phi}_i^u(b_j)$. If $\phi_i^u(b) \in C_{b_j}$ then $\phi_i^u(b_j) = \phi_i^u(b)$ as desired. Otherwise, $\phi_i^u(b_j)$ cannot be closer to u than $\phi_i^u(b)$, satisfying Condition ($\star$) from the definition of $\boldsymbol{\Phi}_i^u(b_j)$.

  Regarding Property (B′), consider a vertex $v_1 \in C_{b_j}$ anchored to a vertex $v_2 = \alpha(v_1) \notin C_{b_j}$ and a vertex $u \in S_b$ on the shortest path from v_1 to v_2. The objective is to show that $v_2 = \rho^u_{\ell(v_1)-1}(b_j)$. As α is an anchoring of minimum cost that respects Property (B′), we must have $v_2 = \text{closest}_u\{\rho^u_{\ell(v_1)-1}(b), \phi^u_{\ell(v_1)-1}(b)\}$. If $\phi^u_{\ell(v_1)-1}(b) \in C_{b_j}$, we must have $v_2 = \rho^u_{\ell(v_1)-1}(b)$ as $v_2 \notin C_{b_j}$. By definition of $\phi^u_{\ell(v_1)-1}(b)$ and $\boldsymbol{\rho}^u_{\ell(v_1)-1}(b_j)$, we obtain $\boldsymbol{\rho}^u_{\ell(v_1)-1}(b_j) = \rho^u_{\ell(v_1)-1}(b) = v_2$. If $\phi^u_{\ell(v_1)-1}(b) \notin C_{b_j}$, then we get by definition $v_2 = \boldsymbol{\rho}^u_{\ell(v_1)-1}(b_j)$.

  For Property (C′), consider i and $u,v \in S_{b_j}$, we know that $\mathrm{d}(u,\rho_i^u(b)) \leq \mathrm{d}(u,\rho_i^w(b))$ as (ℓ,α) belongs to $\mathcal{P}(b,\rho(b),\phi(b))$, so we deduce by the definition of $\rho_i(b)$ that $\mathrm{d}(u,\rho_i^u(b_j)) \leq \mathrm{d}(u,\rho_i^w(b_j))$.
- *Forget nodes:* Recall that for a forget node b with respect to v, we have $S_{b_1} = S_b \cup \{v\}$ and $C_{b_1} = C_b \setminus \{v\}$. Let $i_v = \ell(v)$, which can be ∞. Using the fact that $\phi(b)$ satisfies Property (A′), it is easy to see that $i_v \in \boldsymbol{I}_v$. Clearly, the cost to anchor v in ℓ is equal to c_{i_v}.

  For a given i and $u \in S_b$ (so $u \neq v$), assume first that $\phi_i^u(b) \neq v$. Then, $\phi_i^u(b) \in C_{b_1}$ so we must have $\phi_i^u(b_1) = \phi_i^u(b) \in \boldsymbol{\Phi}_i^u(b_1)$. If $\phi_i^u(b) = v$, then we must have $\mathrm{d}(u,v) \leq \mathrm{d}(u,\phi_i^u(b_1))$ as $\alpha \in P(b,\rho(b),\phi(b))$, so $\phi_i^u(b_1) \in \boldsymbol{\Phi}_i^u(b_1)$.

  We now want to show Property (B′). Consider a vertex $v_1 \in C_{b_1}$ of label $\ell(v_1)$ anchored to $\alpha(v_1) = v_2 \notin C_{b_1}$. If $v_2 = v$, then $v_2 = \rho^v_{i_v}(b_1)$ so the property holds for this case. If $v_2 \neq v$, there $v_2 \notin C_b$, so there exists $u \in S_b$ such that $v_2 = \rho^u_{\ell(v_1)-1}(b)$ as α belongs to $\mathcal{P}(b,\rho(b),\phi(b))$. By definition, if $\rho^u_{\ell(v_1)-1}(b) \neq \rho^u_{\ell(v_1)-1}(b_1)$ then $\rho^u_{\ell(v_1)-1}(b_1) = v$ and $\mathrm{d}(u,v) < \mathrm{d}(u,\rho^u_{\ell(v_1)-1}(b_1))$, which contradicts the fact that α is a minimum cost anchoring.

  Property (C′) follows from the definition of $\rho(b_1)$ and the fact that α belongs to $\mathcal{P}(b,\rho(b),\phi(b))$.

  We therefore obtain that $A[b,\rho(b),\phi(b)] = c_{i_v} + A[b_1,\rho(b_1),\phi(b_1)]$, which proves the inequality.
- *Introduce nodes:* If b is an introduce node with respect to v, it has one child b_1 with $S_{b_1} = S_b \setminus \{v\}$ and $C_{b_1} = C_b$. Since $(\ell,\alpha) \in P(b,\rho(b),\phi(b))$, for all i, we have $\phi_i^v(b) =$

$\text{closest}_v\{\phi_i^u(b) \mid u \in S_{b_1}\}$ by Property (A′). Thus, the conditions that would cause $\bar{A}[b, \rho(b), \phi(b)]$ to be infinite are not met.

Consider a vertex $v_1 \in C_{b_1}$ of label $\ell(v_1)$ anchored to $\alpha(v_1) = v_2 \notin C_{b_1}$. There exists $u \in S_b$ such that $v_2 = \rho^u_{\ell(v_1)-1}(b)$ as α belongs to $\mathcal{P}(b, \rho(b), \phi(b))$. If $u \neq v$, we have the result. If $u = v$, then there exists a vertex $w \in S_{b_1}$ on the shortest path from v_1 to v. By Property (C′) for $\rho(b)$, we know that $\mathrm{d}(w, \rho^w_{\ell(v_1)-1}(b)) \leq \mathrm{d}(w, \rho^v_{\ell(v_1)-1}(b))$. As α is a minimum cost anchoring, this inequality must be an equality, so we obtain Property (B′).

Property (C′) holds for $\rho(b_j)$ as it is a subset of $\rho(b)$.

Second direction, $A[b, \rho(b), \phi(b)] \leq \bar{A}[b, \rho(b), \phi(b)]$. Consider a bag b and values $\rho(b)$ and $\phi(b)$. If b is a leaf, the inequality holds as both sides are zero. If $\bar{A}[b, \rho(b), \phi(b)]$ is infinite, the inequality holds as well. We therefore consider the remaining cases. In particular, Property (C′) is respected regarding $\rho(b)$ and b has either one child b_1 or two children b_1 and b_2. Consider, for $j = 1$ or $j \in \{1, 2\}$, the anchoring guarantees $\phi_i^u(b_j) \in \boldsymbol{\Phi}_i^u(b_j)$ and the LAPs (ℓ_j, α_j) on C_{b_j} yielding the value $\bar{A}[b, \rho(b), \phi(b)]$.

Define (ℓ, α) to be the union of these LAPs in case of a join node. If b is a forget node, then extend (ℓ, α) to v, by choosing the label $\ell(v) = i_v \in \boldsymbol{I}_v$ (possibly ∞) which minimizes $\bar{A}[b, \rho(b), \phi(b)]$, as well as $\alpha(v) = v$ if $i_v = \infty$ and $\alpha(v) = \text{closest}_v\{\phi^v_{i_v-1}(b_1), \rho^v_{i_v-1}(b_1)\}$ otherwise. We will show that this (ℓ, α) belongs to $\mathcal{P}(b, \rho(b), \phi(b))$. Then, by definition of $\bar{A}[b, \rho(b), \phi(b)]$, the inequality holds.

- *Join node:* Consider a join node b with children b_1, b_2. Since $C_{b_1} \cap C_{b_2} = \emptyset$, the union of the LAPs is well defined.

  Consider the anchoring guarantee $\phi_i^u(b)$, for some $u \in S_b$. We want to show that $\ell(\phi_i^u(b)) = i$ and that no vertex of C_b of label i is closer to u. If $\phi_i^u(b) \in C_{b_1}$, then by definition, $\phi_i^u(b_1) = \phi_i^u(b)$. Therefore, $\ell(\phi_i^u(b)) = i$ and no vertex in C_{b_1} closer to u is labeled i. We also know that $\mathrm{d}(u, \phi_i^u(b)) \leq \mathrm{d}(u, \phi_i^u(b_2))$, by Condition $(\star)$ in the definition of $\boldsymbol{\Phi}_i^u(b_1)$. Therefore, by symmetry, Property (A′) holds for $\phi_i^u(b) \in C_{b_2}$ as well.

  Regarding Property (B′), consider a vertex v_1 of any C_{b_j} anchored to $v_2 = \alpha(v_1) \notin C_b$. Then, there exists some $u \in S_b$ such that $v_2 = \rho^u_{\ell(v_1)-1}(b_j)$ by definition of $\mathcal{P}(b_j, \rho(b_j), \phi(b_j))$, and as $v_2 \notin C_b$, we must have $v_2 = \rho^u_{\ell(v_1)-1}(b)$.

- *Forget node:* Let b be a forget node with respect to v. That is $S_{b_1} = S_b \cup \{v\}$ and $C_{b_1} = C_b \setminus \{v\}$. Clearly, (ℓ, α) is well defined.

  Consider i and $u \in S_b$ (so $u \neq v$), and assume first that $\phi_i^u(b) \neq v$. Then, $\phi_i^u(b) = \phi_i^u(b_1)$, so $\phi_i^u(b)$ is the closest vertex to u of label i in C_{b_1}. In order to get the condition of Property (A′) for this case, it remains to show that, if $i = i_v$, we have $\mathrm{d}(u, v) \geq \mathrm{d}(u, \phi_i^u(b))$, which follows from the definition of $\boldsymbol{I}_v$. Assume now that $\phi_i^u(b) = v$. By the definition of $\boldsymbol{\Phi}_i^u(b_1)$ in this case, we have $\mathrm{d}(u, \phi_i^u(b_1)) \geq \mathrm{d}(u, v)$. Then, by the definition of $\mathcal{P}(b_j, \rho(b_j), \phi(b_j))$, there is no vertex in C_{b_1} of label i closer to u than v. This implies that $\phi_i^u(b) = v$ is the closest vertex to u of label i in C_b.

  We now prove Property (B′). Consider a vertex v_1 of C_{b_1} (so $v_1 \neq v$) of label $\ell(v_1)$ anchored to $v_2 = \alpha(v_1) \notin C_b$. There exists some $u \in S_{b_1}$ such that $v_2 = \rho^u_{\ell(v_1)-1}(b_1)$ by definition of $\mathcal{P}(b_1, \rho(b_1), \phi(b_1))$ and $v_2 \notin C_b$ so $v_2 \neq v$. Therefore, by definition of $\boldsymbol{\rho}^u_{\ell(v_1)-1}(b_1)$, there must exist some $w \in S_b$ such that $v_2 = \rho^w_{\ell(v_1)-1}(b)$. Consider now v. If $i_v = \infty$ (which can only be the case if $v \notin \mathcal{X}$), we defined its anchor to be $\alpha(v) = v$. If $i_v \neq \infty$, we defined $\alpha(v) = \text{closest}_v\{\phi^v_{i_v-1}(b_1), \rho^v_{i_v-1}(b_1)\}$. If $\alpha(v) \notin C_b$, then $\alpha(v) = \text{closest}_v\{\rho^u_{i_v-1}(b) \mid u \in S_b\}$ by definition of $\boldsymbol{\rho}^v_{i_v-1}(b_1)$, which completes the proof of Property (B′).

– *Introduce node:* Consider an introduce node b with respect to v. That is b has one child b_1 such that $S_{b_1} = S_b \setminus \{v\}$, $C_{b_1} = C_b$. Again, (ℓ, α) is obviously well defined. For each $i < k$ and $u \in S_{b_1}$, we have $\phi_i^u(b) = \phi_i^u(b_1)$, and we have $\phi_i^v(b) \neq \text{closest}_v\{\phi_i^u(b) \mid u \in S_{b_1}\}$. As any path between v and a vertex in S_b contain a vertex in S_{b_1}, Property (A′) is satisfied.

  Consider a vertex v_1 of C_b of label $\ell(v_1)$ anchored to $v_2 = \alpha(v_1) \notin C_b$. There exists some $u \in S_{b_1}$ such that $v_2 = \rho^u_{\ell(v_1)-1}(b_1)$ by definition of $\mathcal{P}(b_1, \rho(b_1), \phi(b_1))$. As $S_{b_1} \subset S_b$, Property (B′) holds.

Thus, for all types of bags, (ℓ, α) is a member of $\mathcal{P}(b, \rho(b), \phi(b))$. Since $\bar{A}[b, \rho(b), \phi(b)]$ gives the minimum cost of all such LAPs, the second inequality holds as well.

Note that every vertex not in the root bag b_r are *forgotten* exactly once, therefore the price to anchor a vertex not in the root bag is paid exactly once in every DP cell indexed by b_r. Vertices in the root bag are never forgotten, so their anchoring cost has not been incurred yet.

Let b_r be the root of T_G, which contains the root r of the k-hop MŠT. Assume we are given a partial labeling $\bar{\ell}$ on $S_{b_r} \setminus \{r\}$ and values $\phi(b_r)$. For all $v \in S_{b_r}$, we define $\rho_i^v(b_r)$ as the vertex w closest to v for which $\bar{\ell}(w) = i$. The minimum cost of a LAP extending $\bar{\ell}$ that respects $\phi(b_r)$ is equal to

$$A[b_r, \rho(b_r), \phi(b_r)] + \sum_{v \in S_{b_r} \setminus \{r\}} \mathrm{d}(v, \text{closest}_v(\phi^v_{\bar{\ell}(v)-1}(b_r), \rho^v_{\bar{\ell}(v)-1}(b_r))).$$

We then pick the $\bar{\ell}$ and ϕ that minimize this value, and obtain a k-hop MŠT.

□

6 Metrics of Bounded Highway Dimension

In our final section, we make use of our result for k-hop MŠT on metrics of bounded treewidth to obtain an approximation scheme for the more general metrics of bounded highway dimension.

Denoting $B_r(v) = \{u \in V \mid \mathrm{d}(u, v) \leq r\}$, the highway dimension of a graph is defined as follows in [15].

Definition 2. *The* highway dimension *of a graph G is the smallest integer h for which there exists some universal constant $c \geq 4$ 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 $B_{cr}(v)$.*

We now restate and prove Theorem 3.

Theorem 3. *For a metric induced by a graph of bounded highway dimension and 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.*

Proof. As proved in [15, Theorem 8.1], six conditions are required to compute a $(1+\varepsilon)$-approximation in quasi-polynomial time for a given problem. It therefore remains to verify that the k-hop MŠT problem verifies them, 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. Two conditions refer to a δ-net of the graph G, which is defined as a subset U of V such that for all $u \in V$, there exists $v \in U$ with $\mathrm{d}(u, v) \leq \delta$ and for all $u, v \in U$, we have $\mathrm{d}(u, v) > \delta$. The conditions and the explanation of why they are fulfilled are detailed below.

- An optimum solution of k-hop MŠT can be computed in time $n^{O(\omega)}$ for a graph of treewidth ω, this is proved in Theorem 2.
- A constant-approximation of k-hop MŠT on metric graphs can be computed in polynomial time, this follows from [21].
- The diameter of the graph can be assumed to be $O(n \cdot OPT_k)$. Indeed, edges of cost larger than $1.52 \cdot 9^{k-2} \cdot OPT_k$ can be deleted after computing the constant-factor approximation of OPT_k as designed in [21].
- An optimum solution on a δ-net U has cost at most $OPT_k + O(n\delta)$. 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.
- The objective function is linear in the edge cost.
- 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 $O(n\delta)$. This procedure 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 [15]. 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. □

Acknowledgments. We thank Jiří Sgall for fruitful discussions on the k-hop MST problem in path metrics.

References

1. Abraham, I., Fiat, A., Goldberg, A.V., Werneck, R.F.: Highway dimension, shortest paths, and provably efficient algorithms. In: Proceedings of the twenty-first annual ACM-SIAM symposium on Discrete Algorithms. pp. 782–793 (2010)
2. Alfandari, L., Paschos, V.: Approximating minimum spanning tree of depth 2. International Transactions in Operational Research **6**(6), 607–622 (1999)
3. Althaus, E., Funke, S., Har-Peled, S., Könemann, J., Ramos, E.A., Skutella, M.: Approximating k-hop minimum-spanning trees. Oper. Res. Lett. **33**(2), 115–120 (2005)
4. Angel, O., Flaxman, A.D., Wilson, D.B.: A sharp threshold for minimum bounded-depth and bounded-diameter spanning trees and steiner trees in random networks. Combinatorica **32**(1), 1–33 (2012)
5. Arya, S., Das, G., Mount, D.M., Salowe, J.S., Smid, M.: Euclidean spanners: Short, thin, and lanky. In: Proceedings of the Twenty-Seventh Annual ACM Symposium on Theory of Computing. p. 489–498 (1995)
6. Balakrishnan, A., Altinkemer, K.: Using a hop-constrained model to generate alternative communication network design. INFORMS Journal on Computing **4**(2), 192–205 (1992)
7. Bar-Ilan, J., Kortsarz, G., Peleg, D.: Generalized submodular cover problems and applications. Theor. Comput. Sci. **250**(1-2), 179–200 (2001)
8. Bern, M., Plassmann, P.: The steiner problem with edge lengths 1 and 2. Information Processing Letters **32**(4) (1989)
9. Carmi, P., Chaitman-Yerushalmi, L., Trabelsi, O.: Bounded-hop communication networks. Algorithmica **80**(11), 3050–3077 (2018)
10. Chimani, M., Mutzel, P., Zey, B.: Improved steiner tree algorithms for bounded treewidth. Journal of Discrete Algorithms **16**, 67–78 (2012)
11. Chimani, M., Spoerhase, J.: Network Design Problems with Bounded Distances via Shallow-Light Steiner Trees. In: 32nd International Symposium on Theoretical Aspects of Computer Science (STACS 2015). vol. 30, pp. 238–248 (2015)
12. Cygan, M., Fomin, F.V., Kowalik, Ł., Lokshtanov, D., Marx, D., Pilipczuk, M., Pilipczuk, M., Saurabh, S.: Parameterized algorithms, vol. 3 (2015), Chapter 7.

13. Dahl, G., Gouveia, L., Requejo, C.: On formulations and methods for the hop-constrained minimum spanning tree problem. In: Handbook of Optimization in Telecommunications, pp. 493–515 (2006)
14. Elkin, M., Solomon, S.: Narrow-shallow-low-light trees with and without steiner points. SIAM Journal on Discrete Mathematics **25**(1), 181–210 (2011)
15. Feldmann, A.E., Fung, W.S., Könemann, J., Post, I.: A $(1+\varepsilon)$-embedding of low highway dimension graphs into bounded treewidth graphs. SIAM Journal on Computing **47**(4), 1667–1704 (2018)
16. Gouveia, L.: Using the miller-tucker-zemlin constraints to formulate a minimal spanning tree problem with hop constraints. Comput. Oper. Res. **22**(9), 959–970 (1995)
17. Gouveia, L., Simonetti, L., Uchoa, E.: 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)
18. Guha, S., Khuller, S.: Greedy strikes back: Improved facility location algorithms. Journal of Algorithms **31**(1), 228 – 248 (1999)
19. Haenggi, M.: Twelve reasons not to route over many short hops. VTC2004-Fall **5**, 3130–3134 Vol. 5 (2004)
20. Hajiaghayi, M.T., Kortsarz, G., Salavatipour, M.R.: Approximating buy-at-bulk and shallow-light k-steiner trees. Algorithmica **53**(1), 89–103 (2009)
21. Kantor, E., Peleg, D.: Approximate hierarchical facility location and applications to the bounded depth steiner tree and range assignment problems. J. Discrete Algorithms **7**(3), 341–362 (2009)
22. Korte, B., Nešetřil, J.: Vojtěch Jarník's work in combinatorial optimization. Discrete Mathematics **235**(1-3), 1–17 (2001)
23. Kortsarz, G., Peleg, D.: Approximating shallow-light trees. In: Proceedings of the Eighth Annual ACM-SIAM Symposium on Discrete Algorithms. p. 103–110 (1997)
24. Laue, S., Matijević, D.: Approximating k-hop minimum spanning trees in euclidean metrics. Inf. Process. Lett. **107**(3-4), 96–101 (2008)
25. Li, S.: A 1.488 approximation algorithm for the uncapacitated facility location problem. Information and Computation **222**, 45 – 58 (2013)
26. Manyem, P., Stallmann, M.F.: Some approximation results in multicasting. Tech. rep. (1996)
27. Saksena, V.R.: Topological analysis of packet networks. IEEE Journal on Selected Areas in Communications **7**(8), 1243–1252 (1989)
28. Voß, S.: The steiner tree problem with hop constraints. Annals OR **86**, 321–345 (1999)