Viewing the Rings of a Tree: Minimum Distortion Embeddings into Trees

Amir Nayyeri* Benjamin Raichel†

Abstract

We describe a $(1+\varepsilon)$ approximation algorithm for finding the minimum distortion embedding of an n-point metric space, (X, d_X), into a tree with vertex set X. The running time of our algorithm is

$$n^2 \cdot \left(\frac{\Delta}{\varepsilon}\right)^{\log(1/\varepsilon)\cdot(1/\varepsilon)^{\lambda+1}\cdot(O(\delta_{opt}))^{2\lambda}}$$

which is parameterized with respect to the spread of X, denoted by Δ, the minimum possible distortion for embedding X into any tree, denoted by δ_{opt}, and the doubling dimension of X, denoted by λ. Hence we obtain a PTAS, provided that δ_{opt} is a constant and X is a finite doubling metric space with polynomially bounded spread, for example, a point set with polynomially bounded spread in constant dimensional Euclidean space. Our algorithm implies a constant factor approximation with the same running time when Steiner vertices are allowed.

Moreover, we describe a similar $(1+\varepsilon)$ approximation algorithm for finding a tree spanner of (X, d_X) that minimizes the maximum stretch. The running time of our algorithm stays the same, except that δ_{opt} must be interpreted as the minimum stretch of any spanning tree of X. Finally, we generalize our tree spanner algorithm to a $(1+\varepsilon)$ approximation algorithm for computing a minimum stretch tree spanner of a weighted graph with a given upper bound deg on the maximum degree, where the running time is parameterized with respect to deg, in addition to the other parameters above. In particular, we obtain a PTAS for computing minimum stretch tree spanners of weighted graphs, with polynomially bounded spread, constant doubling dimension, and constant maximum degree, when a tree spanner with constant stretch exists.

1 Introduction

Given a general metric space (X, d_X), consider the problem of finding a host metric space from within some class of "simple" metric spaces that (X, d_X) can be embedded into while preserving pairwise distances as much as possible. This is a central problem in the algorithmic study of metric spaces, as naturally finding such a simpler metric can unlock a set of efficient algorithmic tools which may be less effective on more complex spaces.

To quantify the extent to which an embedding preserves distances, we consider the (multiplicative) distortion, which is a widely used and studied measure, having many nice properties such as

*School of Electrical Engineering and Computer Science; Oregon State University; nayyeria@eecs.oregonstate.edu; http://web.engr.oregonstate.edu/~nayyeria/. Work on this paper was partially supported by NSF CRII Award 1566624.

†Department of Computer Science; University of Texas at Dallas; Richardson, TX 75080, USA; benjamin.raichel@utdallas.edu; http://utdallas.edu/~bar150630/. Work on this paper was partially supported by NSF CRII Award 1566137.

scale invariance. Formally, given metric spaces (X, d_X) and (Y, d_Y), an *embedding* of X into Y is an injective map $f : X \to Y$, with *expansion* e_f and *contraction* c_f defined as

$$e_f = \max_{\substack{x,x' \in X \\ x \neq x'}} \frac{d_Y(f(x), f(x'))}{d_X(x, x')}, \quad c_f = \max_{\substack{x,x' \in X \\ x \neq x'}} \frac{d_X(x, x')}{d_Y(f(x), f(x'))}.$$

The *distortion* of f is then defined as $\delta_f = e_f \cdot c_f$. Low distortion embeddings have been extensively studied and have been used in a variety of computer science applications (see [IM04, Ind01, Mat13]).

Among alternatives, one of the most widely studied classes of simpler host metric spaces is the class of weighted trees, whose structure is well understood and readily allows one to apply tools such as dynamic programming. Furthermore, such embeddings have found natural applications, for example, in estimating the phylogenetic tree [KW99].

Closely related to finding embeddings into tree metrics of minimum distortion is the problem of finding tree spanners with minimum stretch. Given a graph G, a tree spanner with stretch δ is a spanning tree of G that preserves distances up to a multiplicative factor of δ, or in other words, it is a δ distortion embedding into a spanning tree of G. As minimal distance preserving structures, tree spanners have found applications in distributed systems [DH98, PR01].

In this paper, we describe approximation algorithms for finding low distortion embeddings into trees, and for finding minimum stretch tree spanners. We start with a brief review of related work.

Embedding into trees. Nearly a half century ago, Buneman studied the problem of reconstructing trees from distance measures [Bun71]. He showed that an embedding with distortion one can be found in $O(n^4)$ time if it exists. Later, Agarwala et al. [ABF+99] showed that in the absence of a perfect embedding, finding a minimum distortion embedding is not only NP-complete, but actually APX-hard.[1] Moreover, in certain cases much stronger hardness results are known. For example, finding the minimum distortion embedding into the real line, that is a tree of maximum degree two, is hard to approximate within a polynomial factor even when embedding from weighted tree metrics with polynomial spread [BCIS05] (note the problem is much easier for additive distortion, as there is a 2-approximation [HIL98]). Thus it is natural to consider restrictions on the source metric space. In particular, Bădoiu et al. [BIS07] showed that the minimum distortion embedding for *unweighted* graphs into trees can be approximated within a constant factor in polynomial time. Their result leads to the state of the art 6-approximation algorithm after a couple of improvements [BDH+08, CDN+10]. In contrast to unweighted graphs, far less is known about embedding general metrics into trees. In fact, the only non-trivial approximation algorithm, found by Bădoiu et al. [BIS07], gives an embedding with distortion $(\delta_{opt} \log n)^{\sqrt{\log \Delta}}$, where δ_{opt} is the minimum distortion.[2]

Tree spanners. The history of tree spanner algorithms is somewhat similar. Cai and Corneil initiated the study of tree spanners [CC95], and showed that the 1-spanner of a weighted graph, if it exists, coincides with the minimum spanning tree, and therefore can be computed efficiently. Nevertheless, computing t-spanners is NP-complete for $t > 1$. For *unweighted* graphs, in the same paper it was shown that the situation is slightly better: polynomial time algorithms exist to find

[1] Note [ABF+99] states the additive distortion case is APX-hard, however, Chepoi et al. [CDN+10] noted that the proof also implies the same for the multiplicative distortion for a smaller constant.

[2] There is a different line of research for embedding a graph into a *given* tree (or graph), see references [KRS04, FFL+13, NR17], for some examples. We emphasize that the goal of this paper is different as here we look for the best possible tree to embed into. Also, note that in this paper we focus on multiplicative distortion. See references [HIL98, ABF+99] for results concerning *additive* distortion.

1-spanners or 2-spanners, if they exist, while finding t-spanners is NP-complete for $t > 4$. For unweighted graphs, Emek and Peleg [EP08] show an $O(\log n)$-approximation algorithm for finding minimum stretch tree spanners. More recently, Fomin et al. [FGvL11] showed that for constants t and w, t-spanners of treewidth w for bounded degree graphs can be found in polynomial time if they exist [FGvL11] (also see [Pap15]). To the best of the authors' knowledge no approximation algorithm is known for general metric spaces for $t > 1$.

Geometric tree spanners are an interesting special case, where the input is a weighted graph representing the distances between points from a metric space. Not much is known even for this special case. (Note the significance of requiring that the spanner is a tree, as there are many results when other sparse graphs are allowed.) Eppstein [Epp00] asked whether it is possible to compute the minimum stretch geometric tree spanner for a planar point set, either exactly or approximately, in polynomial time. Cheon et al. [CHL07] partially answers this question by showing NP-hardness for the decision problem. Eppstein and Wortman [EW05] give a nearly linear time algorithm to find the minimum stretch star for a planar point set. As for approximation algorithms, no non-trivial approximation algorithm is known even for the case when the input is a planar point set.[3]

1.1 Our results

In this paper, we consider the problems of embedding a general metric space into a tree, and finding the minimum stretch tree spanner of a metric space. We give approximation algorithms whose running times are parameterized with respect to: δ_{opt}, the minimum distortion (or stretch); Δ, the spread of X; and λ, the doubling dimension of X.

In our main theorem, we show an algorithm to embed a general metric space (X, d_X) into a tree with vertex set X.

Theorem 1.1. *Let X be an n-point metric space, with doubling dimension λ and spread Δ. Also, let δ_{opt} be the minimum distortion of any embedding of X into any tree with vertex set X. For any $\varepsilon > 0$, there is an*

$$n^2 \cdot \left(\frac{\Delta}{\varepsilon}\right)^{\log(1/\varepsilon)\cdot(1/\varepsilon)^{\lambda+1}\cdot(O(\delta_{opt}))^{2\lambda}}$$

time algorithm to compute a $(1+\varepsilon)\delta_{opt}$ distortion embedding of X into a tree with vertex set X.

To obtain the above result we first show how to compute a $(1+\varepsilon)$-approximation to the minimum distortion embedding into a tree on vertex set X with bounded degree (which may be of independent interest). The above result is then obtained by arguing that our bounded doubling dimension assumption implies that a tree with arbitrary degree can be embedded into a tree with bounded degree with distortion at most $1 + \varepsilon$.

The result of Gupta [Gup01], which shows that Steiner vertices can help only up to a factor of eight in the distortion (see Lemma 2.3), implies that the output of the algorithm of Theorem 1.1 is a constant factor approximation for embedding into a tree when Steiner vertices are allowed.

Corollary 1.2. *Let X be an n-point metric space, with doubling dimension λ and spread Δ. Also, let δ_{opt} be the minimum distortion of any embedding of X into any tree. For any $\varepsilon > 0$, there is an*

$$n^2 \cdot \left(\frac{\Delta}{\varepsilon}\right)^{\log(1/\varepsilon)\cdot(1/\varepsilon)^{\lambda+1}\cdot(O(\delta_{opt}))^{2\lambda}}$$

[3] In this paper, we look for a tree spanner that minimizes the maximum multiplicative stretch. Different variants of the tree spanner problem have been studied before. We refer the interested reader to Liebchen [LW08] for a list of different tree spanner problems and a survey of corresponding results.

time algorithm to compute an $(8+\varepsilon)\delta_{opt}$ distortion embedding of X into a tree.

Our approach can be adapted to compute tree spanners of finite metric spaces. Here, the input is a finite metric space, and the tree spanner must be chosen from the set of all spanning trees of the complete graph representing the metric space.

Theorem 1.3. *Let X be a metric space with doubling dimension λ and spread Δ. Let δ_{opt} be the minimum possible stretch of any spanning tree of X. For any $\varepsilon > 0$, there is an*

$$n^2 \cdot \left(\frac{\Delta}{\varepsilon}\right)^{\log(1/\varepsilon)\cdot(1/\varepsilon)^{\lambda+1}\cdot(O(\delta_{opt}))^{2\lambda}}$$

time algorithm to compute a $(1+\varepsilon)\delta_{opt}$-tree-spanner.

Note that the above theorem only considers spanning trees from graphs that are metric complete. We can strengthen this result to an algorithm for computing tree spanners for general *weighted graphs*, however, the running time of our algorithm depends on the maximum allowable degree for the tree spanner.

Theorem 1.4. *Let $G = (X, E, w)$ be a weighted graph, and let (X, d_X) be its shortest path metric space. Let λ and Δ denote the doubling dimension and spread of (X, d_X), respectively, and let $\mathsf{deg} > 0$ be some integer. Let δ_{opt} be the minimum possible stretch of any spanning tree of G of maximum degree at most deg. For any $\varepsilon > 0$, there is an*

$$n^2 \cdot \left(\frac{\Delta}{\varepsilon}\right)^{\log(\mathsf{deg}/\varepsilon)\cdot(1/\varepsilon)^{\lambda+1}\cdot(O(\delta_{opt}))^{2\lambda}}$$

time algorithm to compute a $(1+\varepsilon)\delta_{opt}$-tree-spanner with maximum degree at most deg.

Note that Theorem 1.1 gives a PTAS for the minimum distortion embedding of a finite metric space (X, d_X) into a tree on vertex set X, provided that δ_{opt} and λ are constants, and that Δ is polynomially bounded. Under the same set of conditions, Corollary 1.2 gives a constant factor approximation algorithm for embedding X into any tree. Again, under the same conditions, Theorem 1.3 gives a PTAS for computing the minimum stretch geometric tree spanner, and Theorem 1.4 gives a PTAS for computing the minimum stretch bounded degree tree spanner of a weighted graph.

1.2 Overview

At a high level, our algorithm tries to build a nearly optimal embedding by stitching together local views of the optimal embedding at each vertex. Roughly speaking, a local view at a vertex x ('vertex' signifying it is the image of the 'point' x) gives estimates for the locations of the images of all other points in X relative to x in the optimal solution. As we don't actually know the optimal embedding, these local views will have to be guessed, and so must be limited in scope. Now it is already too expensive to guess something about the location of each point in X. So consider any point $y \in X$ and suppose that $d_X(x, y) \approx \delta^s$ for some $s \geq 0$, where δ is the optimal distortion. (Note that since distortion is scale invariant for simplicity we can assume the optimal map is non-contracting with expansion at most δ.) If we know where y maps, then for any point $z \in X$ such that $d_X(y, z) \lesssim \delta^{s-1}$, up to a constant factor of $d_T(x, z)$ we will know where z maps, where $d_T(x, z)$ is the distance between x and z in the image. This is because the map being non-contracting implies $d_T(x, y) \geq d_X(x, y) \approx \delta^s$ and expansion at most δ implies $d_X(y, z) \lesssim \delta^s$. Thus rather than

mapping all points at distance δ^s from x, we first compute a δ^{s-1} net[4] (i.e. a maximal set of points whose pairwise distances are at least δ^{s-1}), and then only map the net points. Since we assumed the doubling dimension of X is bounded, this implies a bound on the number of δ^{s-1} points there are within distance δ^s. To apply this to all points in X, we then bin distances by factors of δ, that is we compute such δ^s nets for all for $s = 0, \ldots, \lceil \log_\delta \Delta \rceil$, and then map the nets.

So now we know we are mapping a sufficiently small number of representatives, but since we don't actually know the optimal tree, where are we actually mapping these representative to? A variant of the above described approach was previously used by the by authors in [NR17] (see also [NR15]), where the authors considered the problem of embedding when the tree structure, but not which points map to which vertices, was known. In that case however, as we actually knew the tree we were mapping to, we could point to a specific location in the tree each net point got (approximately) mapped to, which is no longer possible. Moreover, ultimately we need to stitch together these views. Previously dynamic programming was used over the known tree to do this stitching. However, can we now do dynamic programming over a tree which we don't know?

To remedy this situation, for each net point, since we cannot specify its location, instead we specify its distance from x and which branch of x the net point maps to in the image. So consider a view V_x at some $x \in X$ which contains this information. Now even though we don't know the tree structure, the branch information is sufficient to determine for each point in $y \in X$ which one of x's subtrees (if we imagine x as the root of the tree) y maps into, and this partitioning is sufficient to do dynamic programming. Specifically, we check all possible views at the images of all points in a given subtree to see if they can be the root of that subtree. Determining this involves a recursive condition on y's subtrees, which is handled with dynamic programming, and locally verifying whether the view at x and the view at y can be consistently stitched together, that is do their local views appear to be coming from the same world view.

Due to the partitioning property, it is not hard to argue that this dynamic programming procedure will produce a well defined tree. So suppose we output a tree determined by a set of views which are pairwise consistent across the edges of the tree. The remaining question is what is the distortion of the embedding into the tree? Consider the view at a vertex x. Let y be image of some net point which is seen in this view. Consider the path from x to y in the tree. Now the views along this path are edgewise consistent, meaning that when we walk from x to y, the next edge is always the branch assigned to y in the current view, and moreover as we walk across this edge, the distance to y according the views should change by exactly the length of the edge we traversed. In other words, edgewise consistency suffices to imply any pair of our views (even if not adjacent in the tree) are consistent. Thus as we only guessed views which had distortion at most δ with respect to the center of the view, and since we have a view for each $x \in X$, one can infer that globally, we get an embedding with distortion at most δ.

Unfortunately, this all too good to be true. Guessing the branch information for the views is cheap (if the maximum degree is bounded), however, guessing the distances of each net point exactly is far too expensive. At first blush, the solution may seem obvious. Just record the distances approximately since already we are getting an approximate solution since rather then mapping each point we are mapping its closest net point at an appropriate scale. Specifically, if a view is mapping a scale s net point y, then record the distance from x to y in the image up to a factor of roughly δ^s. This approach however has a fatal flaw. Suppose the view at x claims the distance to y in the tree is in between $10\delta^s$ and $11\delta^s$. As we walk from x to y in the tree, at some point our estimate of the distance to y in the current view will have to be decreased (otherwise we never reach y). The issue is that in our dynamic program, since we don't actually know the tree structure, there is no way

[4]Actually, for technical reasons we end up needing a δ^{s-2} net to obtain our desired high level of accuracy.

to know when this update should happen (and this is a key difference from situation in [NR17]). Specifically, our dynamic program must try both long and short edges on this path. If it tries an edge that is longer than δ^s, well then it knows the estimate must be decreased at the next view. However, if the next edge is much smaller than δ^s then knowing whether to update or not means knowing where in the range $[10\delta^s, 11\delta^s]$ the distance to y lies, i.e. we are back to needing to know the distance exactly. In other words, if we walk down a long path with short edges, the views across each edge look consistent, but by the time we reach y something will have gone wrong.

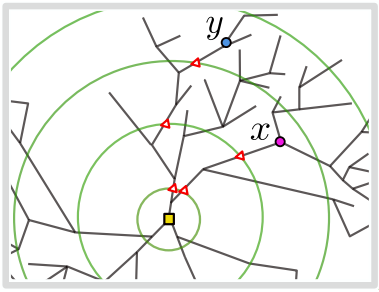

Figure 1: The anchor point is the yellow square, beacon rings are green circles, and beacon ring crossings of the x-to-y path are red triangles.

To resolve this issue, rather than recording the exact distance to the image of each net point, instead we fix an arbitrary vertex $\mathsf{a} \in X$, called the *anchor* point, and only record the distance to a exactly. Note that to check if two views across a given edge in the tree are consistent with respect to the anchor, we just verify that their claimed distances to the anchor differ by exactly the length of the edge. (Note whether the distance should go up or down, depends on whether we are walking towards or away from the anchor, and hence we also record the branch of the anchor.) Now for a given integer $s \leq \lceil \log_\delta \Delta \rceil$, imagine placing a set of concentric rings around the anchor a, with radii $i\delta^s$ for all integers $i \geq 0$ (see Figure 1). Call these *beacon rings* and consider the locations where these rings cross the tree. For a given net point y of scale s whose image is seen in the view V_x at x, we can now record an approximate distance to y, by recording the number of ring crossings that are on the x-to-y path in the tree. Unlike our approximations before which we didn't know how to update, observe that these approximations can be updated precisely, since we know the distance to a exactly, and so it is trivial to detect when an edge crosses one of these rings. As a simple analogy, when driving from point A to point B on the highway, if one records the number of mile markers that get passed, then one will know the distance from A to B, at the resolution of a mile.

Outline. After covering basic background in Section 2, in Section 3 we present our main result for approximating minimum distortion embeddings of metric spaces into bounded degree trees. We then show how to remove the bounded degree assumption in Section 4, by proving that with $(1+\varepsilon)$ distortion, any tree can be embedded into a tree whose degree is bounded by a function only depending on ε and the doubling dimension. Finally, in Section 5, we show that our algorithm can be adapted to find tree spanners by formulating the problem in a more general setting.

2 Preliminaries

Graphs and metrics. We use $G = (V, E, w)$ to denote an undirected graph with vertex set V, edge set E, and positive edge weight function $w : E \to \mathbb{R}^+$. The ***shortest path metric*** (G, d_G) of a graph G is defined by the distance function $d_G : V \times V \to \mathbb{R}^{\geq 0}$, where $d_G(u, v)$ is the length of

the shortest u-to-v path in G. For each $u \in V$, we define $\mathbf{adj_G(u)}$ to be a list of all incident edges to u in G.

Given a metric space (X, d_X), a point $x \in X$, and a radius $r \in \mathbb{R}^+ \cup \{0\}$, the ***ball*** $B(x, r)$ is the subset of all points of X whose distance to x is at most r. The ***doubling dimension*** of a metric space (X, d_X) is the smallest $\lambda \in \mathbb{R}^+$ such that for any $r \in \mathbb{R}^+$, each ball of radius r can be covered by at most 2^λ balls of radius $r/2$. We find the following lemma of Gupta et al. [GKL03] helpful in the analysis in this paper.

Lemma 2.1 (Gupta et al. [GKL03], Proposition 1.1)**.** *Let (X, d_X) be a metric with doubling dimension λ, and let $X' \subseteq X$. If all pairwise distances in X' are at least ℓ, then any ball of radius R in X contains at most $\left(\frac{2R}{\ell}\right)^\lambda$ points of X'.* [5]

Embeddings and distortion. An embedding of a metric space (X, d_X) to a metric space (Y, d_Y) is an injective map $f : X \to Y$. The ***contraction*** c_f and the ***expansion*** e_f of f are defined as

$$c_f = \max_{x,y \in X, x \neq y} \frac{d_X(x, y)}{d_Y(f(x), f(y))}, \quad \text{and} \quad e_f = \max_{x,y \in X, x \neq y} \frac{d_Y(f(x), f(y))}{d_X(x, y)}.$$

An embedding is called ***non-contracting*** if its contraction is at most one. The ***distortion*** of f is defined as $\delta = c_f \cdot e_f$. Often in this paper we consider the identity map as an embedding from a metric space (X, d_X) to the shortest path metric (X, d_T) of a tree $T = (X, E_T, w_T)$. To simplify notation, in these cases, we drop f and compare x-to-y distance in X, denoted by $d_X(x, y)$, with the x-to-y distance in T, denoted by $d_T(x, y)$. Also to simplify, we refer to the identity map (X, d_X) to (X, d_T) as the ***embedding defined by*** T. We use $\delta_{opt}(X)$ to refer to the smallest possible distortion for embedding X into any tree. We use $\delta_{opt}(X, \mathsf{deg})$ to refer to the smallest possible distortion for embedding X into any tree of maximum degree at most deg. Since distortion is scale invariant a non-contracting embedding of expansion $\delta_{opt}(X)$ always exists, and throughout the text we assume we look for a such an embedding.

We found the following lemma helpful when working with embeddings between shortest path metrics of graphs.

Lemma 2.2 (Kenyon et al. [KRS04], Proposition 2.3)**.** *Let $G = (V_G, E_G)$ and $H = (V_H, E_H)$ be two positively weighted undirected graphs, and let d_G and d_H be their shortest path metrics, respectively. Let $f : V_G \to V_H$ be a bijection. Then the expansion of f is achieved by an adjacent pair $u, v \in V_G$, and the contraction of f (or the expansion of f^{-1}) is achieved by an adjacent pair $x, y \in V_H$.*

In this paper, we consider embedding into trees both when Steiner vertices are allowed and when they are not allowed. The following Lemma of Gupta, ensures that the optimal tree metrics for these two problems differ up to a factor of at most eight.

Lemma 2.3 (Gupta [Gup01], Theorem 1.1)**.** *Given a tree $T' = (V', E', w')$ with shortest path metric $d_{T'}$, and a set of required vertices $V \subseteq V'$, there exists a tree $T = (V, E, w)$ with shortest path metric d_T such that for all $x, y \in V$*

$$1 \leq \frac{d_T(x, y)}{d_{T'}(x, y)} \leq 8.$$

Moreover, T can be computed in polynomial time.

[5] Note that λ in their paper is the doubling constant, whereas in this paper it denotes the doubling dimension.

Tree spanners. Let $G = (V, E_G, w_G)$ be a graph, and let $T = (V, E_T, w_T)$ be a spanning tree of G, where w_T is the restriction of w_G to E_T. Let $e = (u, v) \in E_G$. The ***stretch*** (or dilation) of the edge e is defined as

$$str_T(e) = \frac{d_T(u, v)}{w_G(u, v)}.$$

Note that $d_T(u, v) \geq d_G(u, v)$. The ***stretch*** (or dilation) of T is then defined as the maximum stretch of the edges in E_G, $str_T = \max_{e \in E_G} str_T(e)$. By Lemma 2.2, the identity map is a map of distortion str_T from (V, d_G) to (V, d_T). If $str_T = t$, we say that T is a ***t-tree spanner*** of G. Hence, finding the minimum stretch spanning tree is equivalent to finding the spanning tree into which the identity map has the lowest distortion.

Let (X, d_X) be a metric space, and let $T = (X, E_T, w_T)$, where w_T is the restriction of d_X to E_T. Let $x, y \in X \times X$. The ***geometric stretch*** (or dilation) of the pair (x, y) is defined as

$$str_T(x, y) = \frac{d_T(x, y)}{d_X(x, y)}.$$

Again, note $d_T(x, y) \geq d_X(x, y)$. The ***geometric stretch*** of T is then defined as

$$str_T = \max_{x,y \in X \ \& \ x \neq y} str_T(x, y).$$

If $str_T = t$, we say that T is a ***geometric t-tree spanner*** of X. Sometime, when it is clear from the context, we drop the adjective geometric.

3 Bijective embedding into trees

In this section, we consider the problem of embedding a metric space (X, d_X), with doubling dimension λ and spread Δ into a weighted tree $T = (X, E_T, w_T)$ *with vertex set* X (i.e. defining a bijection), and maximum degree deg. We use $\delta_{opt}(X, \mathsf{deg})$ to denote the minimum achievable distortion of such an embedding. We show a $(1 + \varepsilon)$-approximation algorithm for finding this optimal embedding (Theorem 3.13). Theorem 1.1 is then immediately implied from Theorem 3.13 and Corollary 4.3.

3.1 Setup

During this section assume that the smallest distance in X is one, so the largest distance is Δ. This can be ensured by scaling X. Let $\delta \geq \delta_{opt}(X, \mathsf{deg})$, and let a be an arbitrary fixed point of X, called the ***anchor***. Let $\{x_0\} = X_{\geq S+1} \subseteq X_{\geq S} \subseteq \ldots \subseteq X_{\geq 0} = X$, where $X_{\geq s}$ is a maximal subset of points with mutual distances at least δ^s, $S = \lceil \log_\delta(\Delta) \rceil$, and $x_0 \in X$ is an arbitrary point. For technical reasons we extend these sets to be defined for negative values of s, where $X_{\geq s} = X$ for any negative value s. A point $x \in X$ is called a ***scale s point*** if $X_{\geq s}$ is the sparsest net in the sequence that contains x. The ***scale s nearest neighbor*** of a point $x \in X$ is denoted by $nn_s(x)$, and is defined to be the closest point of $X_{\geq s}$ to x. Note that by the maximality of $X_{\geq s}$, we have that $d_X(x, nn_s(x)) < \delta^s$. Therefore, for example, $nn_s(x) = x$ for all $x \in X$ and for any $s \leq 0$.

We build a tree into which X can be embedded with nearly optimal distortion in this section. Part of the process is to find the edge weights of this tree. The following lemma limits the range of the search space for the weight values, while ensuring a bounded approximation factor.

Lemma 3.1. *Let $G = (V, E, w)$ be a graph with minimum edge weight one, and let d_G be the shortest path metric of G. For any $0 < \sigma \leq 1$, G can be embedded to a graph $G' = (V, E, w')$ whose edge weights are multiples of σ with distortion at most $1 + \sigma$.*

Proof. Let $G' = (V, E, w')$ be the graph obtained from G by setting the weight of each edge e to $w'(e) = \lceil w(e)/\sigma \rceil \cdot \sigma$. We bound the distortion of the identity map from (V, d_G) to $(V, d_{G'})$. Since $w'(e) \geq w(e)$ the map is non-contracting. Furthermore, for each $e = (x, y) \in E$, we have

$$w'(e) \leq w(e) + \sigma \leq w(e) \cdot \left(1 + \frac{\sigma}{w(e)}\right) \leq w(e) \cdot (1 + \sigma),$$

as $w(e)$ is at least one. Hence, by Lemma 2.2, the distortion is at most $1 + \sigma$. □

3.2 Views

The key building blocks used in our algorithm are *views*, which are collections of relevant information about what the embedding looks like around the images of points in X. This information is limited in scope so that it can be guessed by our algorithm. Specifically the view at a vertex x ('vertex' signifying it is the image of the 'point' x), specifies the degree of the image of x, the location of the anchor vertex relative to the image of x, and approximate relative locations of the images of all scale s points that are at distance $O(\delta^s)$ from x in the preimage. To describe the location of the anchor vertex we specify the branch adjacent to vertex x which leads to the anchor as well as the exact distance to the anchor. Similarly, for each vertex y which is the image of one of these scale s points, we specify the branch, but rather than specifying the distance to y exactly we just record the number of beacon ring crossings (as described in the overview) of the x-to-y path in the image.

Formally, a ***view*** $V_x = (\deg_x, (A^b_x, A^d_x), \{(b^s_x, r^s_x)\}_{L \leq s \leq S})$ at $x \in X$ is a tuple with parameters $(\mathsf{deg}, \sigma, c, \delta)$ where each component is defined as follows.

(1) An integer, $\deg_x \in \{1, 2, \ldots, \mathsf{deg}\}$.

(2) (a) $A^b_x \in \{1, 2, \ldots, \deg_x\} \cup \{null\}$.
(b) $A^d_x \in \{0, \sigma, 2\sigma, \ldots, \lfloor \delta\Delta/\sigma \rfloor \cdot \sigma\}$.

(3) For each $L \leq s \leq S$

(a) $b^s_x : X_{\geq s} \cap B(x, c \cdot \delta^{s+2}) \to \{1, \ldots, deg_x\} \cup \{null\}$.
(b) $r^s_x : X_{\geq s} \cap B(x, c \cdot \delta^{s+2}) \to \{0, 1, 2, \ldots, \lfloor c\delta^2 \rfloor\}$.

In the definition above we set $L = \lfloor -(\log_\delta c + 3) \rfloor$, as the minimum distance in X is one, and so $B(x, c \cdot \delta^s)$ is empty for $s \leq L$. Throughout the text c is considered to be a sufficiently large value, which will be specified later, and intuitively acts as a dial controlling the approximation quality of the embedding. Whenever, it is clear from the context, we drop the specification of the parameters to simplify the explanation. We say that a point y is ***visible*** under V_x at scale s, if it is in the domain of b^s_x and r^s_x. We say that a point y is visible under V_x if there exists an $L \leq s \leq S$ such that y is is visible under V_x at scale s.

The first step of our algorithm is to list the set of possible views at every point of X. The following lemma bounds the number of such views.

Lemma 3.2. *There are at most* $\frac{1}{\sigma} \cdot (c\Delta)^{O(\log_\delta(c \cdot \mathsf{deg}))(2c\delta^2)^\lambda}$ *different views at any* $x \in X$*. Moreover, a list of these views can be constructed in time linear in the list size.*

Proof. We enumerate all possibilities for a view $V_x = (\deg_x, (A^b_x, A^d_x), \{(b^s_x, r^s_x)\}_{L \leq s \leq S})$ at x. For the first parameter $\deg_x$ in the tuple, there are at most deg possibilities. For (A^b_x, A^d_x) there are at most $(\mathsf{deg} + 1)(\delta\Delta/\sigma + 1)$ possibilities.

So what remains is to bound the possibilities for the b_x^s and r_x^s functions. For each point $y \in X_{\geq s} \cap B(x, c \cdot \delta^{s+2})$ and for each scale $L \leq s \leq S$, there are at most $\mathsf{deg}+1$ possibilities for $b_x^s(y)$, and at most $c\delta^2+1$ possibilities for $r_x^s(y)$. By Lemma 2.1, there are at most $(2c\delta^{s+2}/\delta^s)^\lambda = (2c\delta^2)^\lambda$ points in $X_{\geq s} \cap B(x, c \cdot \delta^{s+2})$. Therefore, for any $L \leq s \leq S$, there are at most $((\mathsf{deg}+1) \cdot (c\delta^2 + 1))^{(2c\delta^2)^\lambda}$ number of choices for b_x^s and r_x^s.

There are $S - L + 1$ scales overall, so the total number of views at x is at most:

$$\mathsf{deg} \cdot (\mathsf{deg}+1)(\delta\Delta/\sigma + 1) \cdot \left(((\mathsf{deg}+1) \cdot (c\delta^2+1))^{(2c\delta^2)^\lambda}\right)^{\lceil \log_\delta(\Delta)\rceil - \lfloor -(\log_\delta c + 3)\rfloor + 1}$$

$$= \frac{\Delta}{\sigma}\left((\mathsf{deg} \cdot c\delta)^{O(\log_\delta(c\Delta))(2c\delta^2)^\lambda}\right) = \frac{\Delta}{\sigma}\left(\delta^{O(\log_\delta(c\cdot\mathsf{deg})\log_\delta(c\Delta))(2c\delta^2)^\lambda}\right) = \frac{1}{\sigma}(c\Delta)^{O(\log_\delta(c\cdot\mathsf{deg}))(2c\delta^2)^\lambda}. \square$$

Feasibility/Plausibility. From a list of views generated by Lemma 3.2, we would like to only keep the views that can be completed into ***feasible*** trees on X, that is trees that define non-contracting embeddings of expansion at most δ. To formally describe our desired properties for such views, we define restriction of embeddings, and extensions of views as follows.

Let $T = (X, E_T, w_T)$ be a tree, and let $x \in X$. We define the ***restricted*** view of T around x to be the view $V_x = (\deg_x, (A_x^b, A_x^d), \{(b_x^s, r_x^s)\}_{L \leq s \leq S})$ specified as follows.

(1) $\deg_x = \deg_T(x)$, where $\deg_T(x)$ denotes the degree of x in T.

(2) $A_x^d = \lfloor d_T(\mathsf{a}, x)/\sigma \rfloor \cdot \sigma$.

(3) Fix a global ordering on the edges of T, and let $\ell : \mathrm{adj}(x) \to \{1, \ldots, \deg_x\}$ be the bijection that (for every $1 \leq i \leq \deg_x$) assigns the ith element of $\mathrm{adj}(x)$ to i. We have:

   (a) If $x = \mathsf{a}$ then $A_x^b = null$, otherwise $A_x^b = \ell(e)$, where e is the first edge of the x-to-a path in T.

   (b) For each $L \leq s \leq S$ and $y \in X_{\geq s} \cap B(x, c \cdot \delta^{s+2})$,

      (i) If $x = y$ then $b_x^s(y) = null$, otherwise $b_x^s(y) = \ell(e)$, where e is the first edge of the x-to-y path in T.

      (ii) $r_x^s(y) = \lfloor d_T(x, \mathsf{a})/\delta^s \rfloor + \lfloor d_T(y, \mathsf{a})/\delta^s \rfloor$ - $2\lfloor d_T(u, \mathsf{a})/\delta^s \rfloor$, where u is the vertex in T that is closest to a on the path from x to y (i.e. u is the lowest common ancestor of x and y if the root is a).

Note that the restricted view of T around x is uniquely defined for fixed values of a, deg, δ, c, and σ. If V_x is the restricted view of T around x, we say that T is an ***extension*** of V_x. Note that a view can possibly be extended to several different trees. A view is called ***feasible*** if it can be extended to a feasible tree. Such an extension is called a ***feasible extension*** of the view.

Ideally, we would like to be able to disregard all non-feasible views from the lists computed by Lemma 3.2. However, it seems impossible to determine feasibility by merely examining a view in isolation from other views. Fortunately, the following weaker condition on views, which can be tested quickly, suffices for our algorithm. We say that a view V_x is ***plausible*** if for any $L \leq s \leq S$ and any $y \in \mathrm{Dom}(r_x^s)$, we have

$$d_X(x, y) - 2\delta^s \leq r_x^s(y) \cdot \delta^s \leq \delta d_X(x, y) + 2\delta^s.$$

Note radii of successive beacon rings differ by δ^s, and hence the need for the additive factor of $2\delta^s$, as this is longest a shortest path can be without crossing a beacon ring. Moreover, observe that at a sufficiently small scale the additive error in the above definition will become a multiplicative

one. Intuitively a view is plausible if non-feasibility of the view cannot be concluded by examining it in isolation from other views. The following lemma ensures that the plausibility of a view can be checked efficiently.

Lemma 3.3. *There is an $O((2c\delta^2)^\lambda \cdot \log_\delta(c\Delta))$ time algorithm to check the plausibility of a view.*

Proof. Let $V_x = (\deg_x, (A_x^b, A_x^d), \{(b_x^s, r_x^s)\}_{L \le s \le S})$ be a view at $x \in X$. For each $L \le s \le S$, we have $|\mathrm{Dom}(r_x^s)| \le (2c\delta^2)^\lambda$ (by Lemma 2.1). For each element in $\mathrm{Dom}(r_x^s)$ the plausibility condition can be checked in constant time. Therefore, the total running time for checking plausibility is $O((2c\delta^2)^\lambda \cdot (S - L)) = O((2c\delta^2)^\lambda \cdot \log_\delta(c\Delta))$. □

The partition function. Although a view provides information only about the images of a relatively small subset of X, more can be deduced from it. Specifically, a view V_x at x uniquely determines the connected components of $T\backslash\{x\}$ for *every* feasible extension T of V_x (if any exists). Note that a priori it is not even clear that these connected components must be the same in different feasible extensions of V_x.

Lemma 3.4. *Let $V_x = (\deg_x, (A_x^b, A_x^d), \{(b_x^s, r_x^s)\}_{L \le s \le S})$ be any view at $x \in X$. There is an algorithm to compute a partition P of $X\backslash\{x\}$ in $O(n \log_\delta(c\Delta))$ time with the following property.*

- *For every feasible extension T of V_x, and any $y, z \in X\backslash\{x\}$, y and z belong the the same connected component of $T\backslash\{x\}$ if and only if y and z belong to the same set of P.*

Proof. Let $y \in X$, and let s be the smallest scale such that b_x^s and r_x^s act on $nn_s(y)$, where s is well defined as b_x^S acts on all $X_{\ge S}$. We show that y and $nn_s(y)$ must belong to the same connected component of $T\backslash\{x\}$ in any feasible extension T of V_x. Note that since b_x^s acts on $nn_s(y)$, the connected component containing $nn_s(y)$ is specified by V_x, and so the lemma statement will then follow.

By the definition of $nn_s(y)$ and the feasibility of T, we have:

$$d_X(y, nn_s(y)) \le \delta^s \Rightarrow d_T(y, nn_s(y)) \le \delta^{s+1}.$$

Suppose, to derive a contradiction, that y and $nn_s(y)$ belong to different connected components of $T\backslash\{x\}$. That is, the path from y to $nn_s(y)$ in T contains x. Consequently,

$$d_T(x, y) \le d_T(y, nn_s(y)) \le \delta^{s+1}.$$

As T is feasible, it defines a non-contracting embedding. It follows that $d_X(x, y) \le \delta^{s+1}$. So, by the triangle inequality, we have:

$$d_X(x, nn_{s-1}(y)) \le d_X(x, y) + d_X(y, nn_{s-1}(y)) \le \delta^{s+1} + \delta^{s-1} \le 2\delta^{s+1}.$$

Therefore, b_x^{s-1} must act on $nn_{s-1}(y)$ (assuming $c \ge 2$), which is a contradiction with the assumption that s is the smallest scale for which b_x^s acts on $nn_s(y)$. □

3.3 Consistency of views

Ultimately we wish to merge together plausible views at different vertices to yield a feasible tree. To this end, the views that are merged must have consistent descriptions of that tree. As a first step, we define when two plausible views can be merged over an edge.

Let $x, y \in X$. Let $V_x = (\deg_x, (A_x^b, A_x^d), \{(b_x^s, r_x^s)\}_{L \le s \le S})$ and $V_y = (\deg_y, (A_y^b, A_y^d), \{(b_y^s, r_y^s)\}_{L \le s \le S})$ be plausible views at x and y, respectively. Let $i \in \{1, 2, \ldots, \deg_x\}$, and $j \in \{1, 2, \ldots, \deg_y\}$. We say that V_x and V_y can be ***consistently merged*** over (i, j) if the following conditions hold.

(1) Either $A^b_x = i$ or $A^b_y = j$, but not both.

(2) For each $L \le s \le S$, and each $z \in \mathrm{Dom}(b^s_x) \cap \mathrm{Dom}(b^s_y)$ one of the following conditions hold

(a) Either $b^s_x(z) = i$ or $b^s_y(z) = j$, but not both.

(b1) If $b^s_x(z) = i$ and $A^b_x \neq i$ then $r^s_x(z) - r^s_y(z) = \lfloor A^d_y/\delta^s \rfloor - \lfloor A^d_x/\delta^s \rfloor$.

(b2) If $b^s_x(z) = i$ and $A^b_x = i$ then $r^s_x(z) - r^s_y(z) = \lfloor A^d_x/\delta^s \rfloor - \lfloor A^d_y/\delta^s \rfloor$.

(b3) If $b^s_x(z) \neq i$ and $A^b_x = i$ then $r^s_y(z) - r^s_x(z) = \lfloor A^d_x/\delta^s \rfloor - \lfloor A^d_y/\delta^s \rfloor$.

(b4) If $b^s_x(z) \neq i$ and $A^b_x \neq i$ then $r^s_y(z) - r^s_x(z) = \lfloor A^d_y/\delta^s \rfloor - \lfloor A^d_x/\delta^s \rfloor$.

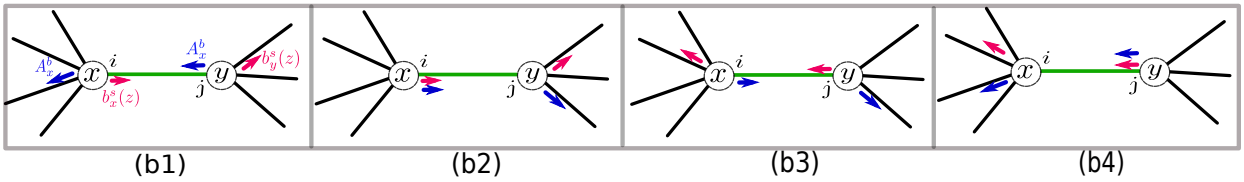

Figure 2: Consistency conditions (2)-(b1) to (2)-(b4).

The following lemma gives an algorithm to check consistency for given V_x, V_y, i, and j.

Lemma 3.5. *Let $V_x = (\deg_x, (A^b_x, A^d_x), \{(b^s_x, r^s_x)\}_{L \le s \le S})$ and $V_y = (\deg_y, (A^b_y, A^d_y), \{(b^s_y, r^s_y)\}_{L \le s \le S})$ be plausible views at x and y, respectively. Let $i \in \{1, 2, \ldots, \deg_x\}$, and $j \in \{1, 2, \ldots, \deg_y\}$. There is an $O((2c\delta^2)^{2\lambda} \cdot \log_\delta(c\Delta))$ time algorithm to check if V_x and V_y can be consistently merged over (i, j).*

Proof. Condition (1) can be checked in $O(1)$ time. For each $L \le s \le S$, we have $|\mathrm{Dom}(b^s_x)| \le (2c\delta^2)^\lambda$ and $|\mathrm{Dom}(b^s_y)| \le (2c\delta^2)^\lambda$. Therefore, their intersection can be computed in $O((2c\delta^2)^{2\lambda})$ time. For each element in the intersection, conditions (a) and (b1) through (b4) can be checked in constant time. Therefore, the total running time for checking condition (2) is $O((2c\delta^2)^{2\lambda} \cdot (S - L)) = O((2c\delta^2)^{2\lambda} \cdot \log_\delta(c\Delta))$. □

Consistent set of views. The images of two points $x, y \in X$ can be adjacent in an optimal tree only if there are plausible views at each of them that can be consistently merged. By merging pairs of plausible views one at a time, our algorithm builds a tree T over X, and an accompanying collection of views for each vertex in X that can be consistently merged over the edges of T. We call such a collection a consistent set over T, and formally define it as follows.

Let $\mathcal{V}$ be a set of plausible views, one at each vertex of X, and let $T = (X, E_T, w_T)$ be a tree. We say that $\mathcal{V}$ is a ***consistent set*** of views over T if there are bijections $\ell_x : \mathrm{adj}(x) \to \{1, 2, \ldots, \deg_x\}$ for all $x \in X$ with the following properties.

(1) For each $e = (x, y) \in E_T$ and the corresponding views $V_x, V_y \in \mathcal{V}$, we have that V_x and V_y are consistent over $(\ell_x(e), \ell_y(e))$, and $w_T(e) = |A^d_x - A^d_y|$.

(2) For each x and its corresponding view $V_x = (\deg_x, (A^b_x, A^d_x), \{(b^s_x, r^s_x)\}_{L \le s \le S})$, the following three conditions are equivalent (i) $A^b_x = null$, (ii) $A^d_x = 0$, and (iii) $x = \mathsf{a}$.

3.4 A consistent set is all we need

Ultimately our algorithm will attempt to grow a nearly optimal tree from the anchor a using dynamic programming. The space of possible trees over X is far too large to be explored. Thus additionally we guess a view at each vertex as we go, as this will severely limit the possibilities for the subtrees. To this end, in this section we show that limiting to the space of trees with such views is valid. That is, if any set of views with parameters $(\mathsf{deg}, \sigma, c, \delta)$ are consistent over a tree T then it implies the distortion of the embedding defined by T is close to δ (how close depends on c and σ, and is specified below). Thus any set of consistent views will suffice. Moreover, we first show, by considering the restriction of any feasible embedding, that at least one consistent set must exist.

Lemma 3.6. *Let (X, d_X) be a metric space, and let $\delta_{opt} = \delta_{opt}(X, \mathsf{deg})$ be the optimal distortion for embedding X into a tree of max degree at most deg. For any $0 < \sigma \leq 1$ and any $c \geq 1$, there exists a set of plausible views $\mathcal{V}$ with parameters $(\mathsf{deg}, \sigma, c, (1+\sigma)\delta_{opt})$, and a tree $T = (X, E_T, w_T)$ of maximum degree deg such that $\mathcal{V}$ is consistent over T.*

Proof. Let $T' = (X, E_{T'}, w_{T'})$ be a tree, into which X can be embedded with distortion δ_{opt}. By Lemma 3.1, there is a tree $T = (X, E_T, w_T)$ whose edge weights are multiples of σ, into which T' can be embedded with distortion $1 + \sigma$. Therefore, X can be embedded into T with distortion $(1 + \sigma) \cdot \delta_{opt}$. Suppose after relabeling the vertices of T that the identity map from (X, d_X) to (X, d_T) has distortion $(1 + \sigma) \cdot \delta_{opt}$. For each $x \in X$, let V_x be the restricted view of this identity map at x in T with parameters deg, σ, c, and $\delta = (1 + \sigma) \cdot \delta_{opt}$. Let $\mathcal{V} = \bigcup_{x \in X} V_x$. We show that $\mathcal{V}$ is a set of consistent views over T.

First, for any x, we show that V_x is plausible. By the definition of restriction we have:

$$r_x^s(y) = \lfloor d_T(x, \mathsf{a})/\delta^s \rfloor + \lfloor d_T(y, \mathsf{a})/\delta^s \rfloor - 2\lfloor d_T(u, \mathsf{a})/\delta^s \rfloor,$$

where u is the closest vertex to a on the x-to-y path. Removing the floors we obtain:

$$\frac{d_T(x, \mathsf{a}) + d_T(y, \mathsf{a}) - 2d_T(u, \mathsf{a})}{\delta^s} - 2 \leq r_x^s(y) \leq \frac{d_T(x, \mathsf{a}) + d_T(y, \mathsf{a}) - 2d_T(u, \mathsf{a})}{\delta^s} + 2$$

The definition of u implies $d_T(x, y) = d_T(x, \mathsf{a}) + d_T(y, \mathsf{a}) - 2d_T(u, \mathsf{a})$, therefore, we obtain:

$$d_T(x, y) - 2\delta^s \leq r_x^s(y) \cdot \delta^s \leq d_T(x, y) + 2\delta^s.$$

Since the embedding into T is feasible, i.e. non-contracting with expansion at most δ, we conclude

$$d_X(x, y) - 2\delta^s \leq r_x^s(y) \cdot \delta^s \leq \delta d_X(x, y) + 2\delta^s.$$

Next, we show the mutual consistency between these sets of restricted views. Let $x, y \in X$, and let $V_x = (\deg_x, (A_x^b, A_x^d), \{(b_x^s, r_x^s)\}_{L \leq s \leq S})$ and $V_y = (\deg_y, (A_y^b, A_y^d), \{(b_y^s, r_y^s)\}_{L \leq s \leq S})$ be restricted views of the identity map around x and y in T, respectively. Also, let ℓ_x and ℓ_y be the labeling functions induced by the restrictions to x and y. Suppose, $e = (x, y) \in E_T$. We show that V_x and V_y can be consistently merged over $\ell_x(e)$ and $\ell_y(e)$ with weight $w_T(e)$. Condition (1) and (2-a) of consistency are implied by the restriction definition, items (3-a) and (3-b-i). It remains to show that conditions (2-b1) through (2-b4) hold. For any $L \leq s \leq S$ and any $z \in \mathrm{Dom}(b_x^s) \cap \mathrm{Dom}(b_y^s)$, we have

$$r_x^s(z) = \lfloor d_T(x, \mathsf{a})/\delta^s \rfloor + \lfloor d_T(z, \mathsf{a})/\delta^s \rfloor - 2\lfloor d_T(u_{x,z}, \mathsf{a})/\delta^s \rfloor,$$

and

$$r_y^s(z) = \lfloor d_T(y, \mathsf{a})/\delta^s \rfloor + \lfloor d_T(z, \mathsf{a})/\delta^s \rfloor - 2\lfloor d_T(u_{y,z}, \mathsf{a})/\delta^s \rfloor,$$

where $u_{x,z}$ is the closest vertex of the x-to-z path to a, and $u_{y,z}$ is the closest vertex of the y-to-z path to a. We consider conditions (2-b1) through (2-b4) (looking at Figure 2 while reading the following cases may help the reader). Let $i = \ell_x(e)$ for the following case analysis.

Case (2-b1) or (2-b3): We have $b^s_x(z) = i$ and $A^b_x \neq i$, or, $b^s_x(z) \neq i$ and $A^b_x = i$. In both cases, we have $u_{x,z} = x$ and $u_{y,z} = y$. Therefore,

$$r^s_x(z) - r^s_y(z) = \lfloor d_T(y, \mathsf{a})/\delta^s \rfloor - \lfloor d_T(x, \mathsf{a})/\delta^s \rfloor = \lfloor A^d_y/\delta^s \rfloor - \lfloor A^d_x/\delta^s \rfloor.$$

Case (2-b2) or (2-b4): we have $b^s_x(z) = i$ and $A^b_x = i$, or, $b^s_x(z) \neq i$ and $A^b_x \neq i$. In both cases, it is implied that $u_{x,z} = u_{y,z}$. Therefore,

$$r^s_x(z) - r^s_y(z) = \lfloor d_T(x, \mathsf{a})/\delta^s \rfloor - \lfloor d_T(y, \mathsf{a})/\delta^s \rfloor = \lfloor A^d_x/\delta^s \rfloor - \lfloor A^d_y/\delta^s \rfloor.$$

In both cases, the last equality holds because weights of T are integer multiples of σ. □

Next, we show that a consistent set over a tree guarantees near optimal distortion. To that end, we need a few definitions and helper lemmas.

Let $T = (X, E_T, w_T)$ be a tree, and let $\gamma = (v_1, \ldots, v_k)$ be a path in T. We say that γ is ***approaching*** if $d_T(v_1, \mathsf{a}) \geq d_T(v_2, \mathsf{a}) \geq \ldots \geq d_T(v_k, \mathsf{a})$. We say that γ is ***departing*** if $d_T(v_1, \mathsf{a}) \leq d_T(v_2, \mathsf{a}) \leq \ldots \leq d_T(v_k, \mathsf{a})$. Finally, we say that γ is ***monotone*** if it is approaching or departing. Note that any (simple) path γ can be decomposed into $\gamma^- \circ \gamma^+$, such that γ^- is approaching and γ^+ is departing. We show that very accurate information can be deduced from the views of two vertices x and y if the path between them in T is approaching or departing.

Lemma 3.7. *Let $\mathcal{V}$ be a consistent set of views over $T = (X, E_T, w_T)$. Let $x, y \in X$, and let V_x and V_y be the views at x and y, respectively. Finally, let γ be the unique x-to-y path in T. If γ is approaching then $d_T(x, y) = A^d_x - A^d_y$, and if γ is departing then $d_T(x, y) = A^d_y - A^d_x$.*

Proof. Let $\gamma = (x = v_1, \ldots, v_k = y)$. Suppose γ is approaching, the other case is similar. We use induction on k to prove the statement. If $k = 1$ then $x = y$, and the statement trivially holds. If $k > 1$, let $t = v_{k-1}$, and let $V_t \in \mathcal{V}$ be the view at t. By the induction hypothesis, $d_T(x, t) = A^d_x - A^d_t$. Since γ is approaching, A^b_t points to the edge (t, y), and thus since V_t and V_y are consistent over the edge (t, y) in T, we have that $d_T(t, y) = w_T(t, y) = A^d_t - A^d_y$. Overall,

$$d_T(x, y) = d_T(x, t) + w_T(t, y) = (A^d_x - A^d_t) + (A^d_t - A^d_y) = A^d_x - A^d_y$$ □

Lemma 3.8. *Let $\mathcal{V}$ be a consistent set of views over $T = (X, E_T, w_T)$. Let $x, y, z \in X$, and let V_x and V_y be the views at x and y, respectively. Finally, let γ be the unique x-to-y path in T. Suppose γ is monotone and z is in the same connected component with either x or y in $T \backslash (\gamma \backslash \{x, y\})$. If a and z belong to the same connected component of $T \backslash (\gamma \backslash \{x, y\})$ then $r^s_x(z) - r^s_y(z) = \lfloor A^d_x/\delta^s \rfloor - \lfloor A^d_y/\delta^s \rfloor$. Otherwise, $r^s_x(z) - r^s_y(z) = \lfloor A^d_y/\delta^s \rfloor - \lfloor A^d_x/\delta^s \rfloor$.*

Proof. Let $\gamma = (x = v_1, \ldots, v_k = y)$. We use induction on k to prove the statement. If $k = 1$ then $x = y$, and the statement trivially holds. If $k > 1$, let $t = v_{k-1}$, and let $V_t \in \mathcal{V}$ be the view at t. Note that in the lemma statement we assume that z is in the connected component of either x or y in $T \backslash (\gamma \backslash \{x, y\})$, and so z is in the connected component of x or t in $T \backslash (\gamma[x, t] \backslash \{x, t\})$ First, consider the case that a and z belong to the same connected component of $T \backslash (\gamma \backslash \{x, y\})$. By the induction hypothesis,

$$r^s_x(z) - r^s_t(z) = \lfloor A^d_x/\delta^s \rfloor - \lfloor A^d_t/\delta^s \rfloor. \tag{1}$$

Since V_t and V_y are consistent over the edge (t, y) in T, and A^b_t and $b^s_t(z)$ are the same, one of conditions (2-b2) or (2-b4) holds. In either case,

$$r^s_t(z) - r^s_y(z) = \lfloor A^d_t/\delta^s \rfloor - \lfloor A^d_y/\delta^s \rfloor.$$

Substituting in Equation (1) we obtain the lemma statement.

Next, consider the case that a and z belong to different connected components of $T\backslash(\gamma\backslash\{x, y\})$. By the induction hypothesis,

$$r^s_x(z) - r^s_t(z) = \lfloor A^d_t/\delta^s \rfloor - \lfloor A^d_x/\delta^s \rfloor. \tag{2}$$

Since V_t and V_y are consistent over the edge (t, y) in T, and A^b_t and $b^s_t(z)$ are different, one of conditions (2-b1) or (2-b3) holds. In either case,

$$r^s_t(z) - r^s_y(z) = \lfloor A^d_y/\delta^s \rfloor - \lfloor A^d_t/\delta^s \rfloor.$$

Substituting in Equation (2) we obtain the lemma statement. □

Next, we show that the distance estimators in the views provide relatively accurate estimations for the distance of visible vertices in T.

Lemma 3.9. *Let $\mathcal{V}$ be a consistent set of views over $T = (X, E_T, w_T)$. Let $x, z \in X$, $V_x \in \mathcal{V}$ be the view at x, and $L \le s \le S$. If z is visible in V_x at scale s then*

$$d_T(x, z) - 4\delta^s \le r^s_x(z) \cdot \delta^s \le d_T(x, z) + 4\delta^s$$

Proof. Let $\gamma = (x = v_1, \ldots, v_k = z)$ be the unique x-to-z path in T. As noted above, γ can be decomposed into two subpaths $\gamma^- = (v_1, \ldots, v_j = y)$, and $\gamma^+ = (y = v_j, v_{j+1}, \ldots, v_k)$ such that γ^- is approaching, and γ^+ is departing. Thus y is the closest point of γ to the anchor point in T. By Lemma 3.7, we have

$$d_T(x, y) = A^d_x - A^d_y, \quad \& \quad d_T(y, z) = A^d_z - A^d_y.$$

Therefore,

$$d_T(x, z) = A^d_z + A^d_x - 2A^d_y. \tag{3}$$

On the other hand, by Lemma 3.8 we know

$$r^s_x(z) - r^s_y(z) = \lfloor A^d_x/\delta^s \rfloor - \lfloor A^d_y/\delta^s \rfloor, \quad \& \quad r^s_y(z) - r^s_z(z) = \lfloor A^d_z/\delta^s \rfloor - \lfloor A^d_y/\delta^s \rfloor.$$

Consequently,

$$r^s_x(z) - r^s_z(z) = \lfloor A^d_x/\delta^s \rfloor + \lfloor A^d_z/\delta^s \rfloor - 2\lfloor A^d_y/\delta^s \rfloor. \tag{4}$$

To obtain the desired statement, we combine Equations (3) and (4), while noting that the definition of plausible views implies $r^s_z(z) \in \{-2, -1, 0, 1, 2\}$ (as $d_X(z, z) = 0$). First we show the upper bound for $r^s_x(z) \cdot \delta^s$.

$$\begin{aligned} r^s_x(z) \cdot \delta^s &= \lfloor A^d_x/\delta^s \rfloor \cdot \delta^s + \lfloor A^d_z/\delta^s \rfloor \cdot \delta^s - 2\lfloor A^d_y/\delta^s \rfloor \cdot \delta^s + r^s_z(z) \cdot \delta^s \\ &\le (A^d_x/\delta^s) \cdot \delta^s + (A^d_z/\delta^s) \cdot \delta^s - 2(A^d_y/\delta^s - 1) \cdot \delta^s + 2\delta^s \\ &= A^d_x + A^d_z - 2A^d_y + 2\delta^s + 2\delta^s \\ &\le d_T(x, z) + 4\delta^s. \end{aligned}$$

Next, we show the lower bound for $r_x^s(z) \cdot \delta^s$.

$$\begin{aligned} r_x^s(z) \cdot \delta^s &= \lfloor A_x^d/\delta^s \rfloor \cdot \delta^s + \lfloor A_z^d/\delta^s \rfloor \cdot \delta^s - 2\lfloor A_y^d/\delta^s \rfloor \cdot \delta^s + r_z^s(z) \cdot \delta^s \\ &\geq (A_x^d/\delta^s - 1) \cdot \delta^s + (A_z^d/\delta^s - 1) \cdot \delta^s - 2(A_y^d/\delta^s) \cdot \delta^s - 2\delta^s \\ &= A_x^d - \delta^s + A_z^d - \delta^s - 2A_y^d - 2\delta^s \\ &\geq d_T(x,z) - 4\delta^s \end{aligned}$$

□

Now, we are ready to bound the distortion of distances on T. First, we show that this distortion is bounded for a pair of vertices if one is visible under the view at the other one.

Lemma 3.10. *Let $\mathcal{V}$ be consistent set of views over $T = (X, E_T, w_T)$, let $x, z \in X$, and let $V_x \in \mathcal{V}$ be the view at x. If z is visible in V_x then*

$$\left(1 - \frac{6}{c}\right) \cdot d_X(x,z) \leq d_T(x,z) \leq \left(1 + \frac{6}{c}\right) \cdot \delta \cdot d_X(x,z).$$

Proof. Note that if $x = z$ the lemma statement trivially holds, thus, assume otherwise. Let $L \leq s \leq S$ be the smallest scale such that z is visible at scale s in V_x. Since z is visible, and V_x is plausible, we have

$$d_X(x,z) - 2\delta^s \leq r_x^s(z) \cdot \delta^s \leq \delta d_X(x,z) + 2\delta^s.$$

Also, by Lemma 3.9,

$$d_T(x,z) - 4\delta^s \leq r_x^s(z) \cdot \delta^s \leq d_T(x,z) + 4\delta^s.$$

Consequently, we have,

$$d_X(x,z) - 6\delta^s \leq d_T(x,z) \leq \delta d_X(x,z) + 6\delta^s. \tag{5}$$

On the other hand, since z is not visible at scale $s-1$, we have

$$c\delta^{s+1} < d_X(x,z) \Rightarrow \delta^s \leq \frac{d_X(x,z)}{c}.$$

Substituting in Equation (5) we obtain

$$\left(1 - \frac{6}{c}\right) \cdot d_X(x,z) \leq d_X(x,z) - 6 \cdot \delta^s \leq d_T(x,z) \leq \delta d_X(x,z) + 6 \cdot \delta^s \leq \left(1 + \frac{6}{c}\right) \cdot \delta \cdot d_X(x,z). \quad \square$$

Finally, we bound the distortion for any pair of vertices, even if they are not visible under each other's views.

Lemma 3.11. *Let $\mathcal{V}$ be a consistent set of views over $T = (X, E_T, w_T)$, and let $x, z \in X$. We have,*

$$\left(1 - \frac{14}{c-1}\right) \cdot d_X(x,z) \leq d_T(x,z) \leq \left(1 + \frac{20}{c-1}\right) \cdot \delta \cdot d_X(x,z).$$

Proof. Note that if $x = z$ the lemma statement trivially holds, thus, assume otherwise. Let s be the smallest scale such that $nn_s(z)$ is visible at V_x, where s is well defined as b_x^S acts on all $X_{\geq S}$. First, we show that $d_X(z, nn_s(z))$ is small compared to $d_X(x,z)$. By the definition of the scale nearest neighbors we have,

$$d_X(z, nn_s(z)) \leq \delta^s \quad \& \quad d_X(z, nn_{s-1}(z)) \leq \delta^{s-1}. \tag{6}$$

Since $nn_{s-1}(z)$ is not visible at V_x we have,

$$d_X(x, nn_{s-1}(z)) \geq c \cdot \delta^{s+1},$$

and therefore,

$$d_X(x,z) \geq d_X(x, nn_{s-1}(z)) - d_X(z, nn_{s-1}(z)) \geq c\delta^{s+1} - \delta^{s-1} \geq (c-1)\delta^{s+1}.$$

Combining with (6) we obtain,

$$\frac{d_X(z, nn_s(z))}{d_X(x,z)} \leq \frac{\delta^s}{(c-1)\delta^{s+1}} \Rightarrow d_X(z, nn_s(z)) \leq \frac{d_X(x,z)}{(c-1)\delta}. \tag{7}$$

Now we use Inequality (7) and Lemma 3.10 to show bounds for the $d_T(x,z)$. By our assumption $nn_s(z)$ is visible in V_x. Furthermore, as $d_X(z, nn_s(z)) \leq \delta^s$, by the definition of nets, $nn_s(z)$ is visible in V_z. First, we show the upper bound.

$$\begin{aligned}
d_T(x,z) &\leq d_T(x, nn_s(z)) + d_T(nn_s(z), z) && \text{(Triangle inequality)} \\
&\leq \left(1 + \frac{6}{c}\right) \cdot \delta \cdot (d_X(x, nn_s(z)) + d_X(nn_s(z), z)) && \text{(Lemma 3.10)} \\
&\leq \left(1 + \frac{6}{c}\right) \cdot \delta \cdot (d_X(x,z) + 2d_X(nn_s(z), z)) && \text{(Triangle inequality)} \\
&\leq \left(1 + \frac{6}{c}\right) \cdot \delta \cdot \left(d_X(x,z) + \frac{2d_X(x,z)}{(c-1)\delta}\right) && \text{(Inequality (7))} \\
&\leq \left(1 + \frac{6}{c}\right) \cdot \delta \cdot \left(1 + \frac{2}{c-1}\right) \cdot d_X(x,z) && \text{(because } \delta \geq 1\text{)} \\
&\leq \left(1 + \frac{20}{c-1}\right) \cdot \delta \cdot d_X(x,z)
\end{aligned}$$

Next, we show the lower bound.

$$\begin{aligned}
d_T(x,z) &\geq d_T(x, nn_s(z)) - d_T(nn_s(z), z) && \text{(Triangle ineq.)} \\
&\geq \left(1 - \frac{6}{c}\right) \cdot d_X(x, nn_s(z)) - \left(1 + \frac{6}{c}\right) \cdot \delta \cdot d_X(nn_s(z), z) && \text{(Lemma 3.10)} \\
&\geq \left(1 - \frac{6}{c}\right) \cdot (d_X(x,z) - d_X(z, nn_s(z))) - \left(1 + \frac{6}{c}\right) \cdot \delta \cdot d_X(nn_s(z), z) && \text{(Triangle ineq.)} \\
&\geq \left(1 - \frac{6}{c}\right) \cdot d_X(x,z) - \left(\left(1 + \frac{6}{c}\right) \cdot \delta + \left(1 - \frac{6}{c}\right)\right) \cdot d_X(nn_s(z), z) \\
&\geq \left(1 - \frac{6}{c}\right) \cdot d_X(x,z) - \left(\left(1 + \frac{6}{c}\right) \cdot \delta + \left(1 - \frac{6}{c}\right)\right) \cdot \frac{d_X(x,z)}{(c-1)\delta} && \text{(Inequality (7))} \\
&\geq \left(1 - \frac{6}{c} - \frac{1}{c-1} - \frac{6}{c(c-1)} - \frac{1}{(c-1)\delta} + \frac{6}{c(c-1)\delta}\right) \cdot d_X(x,z) \\
&\geq \left(1 - \frac{6}{c} - \frac{2}{c-1} - \frac{6}{c(c-1)}\right) \cdot d_X(x,z) && \text{(because } \delta \geq 1\text{)} \\
&\geq \left(1 - \frac{14}{c-1}\right) \cdot d_X(x,z)
\end{aligned}$$

□

3.5 Dynamic programming

In the previous section we defined the notion of a consistent set $\mathcal{V}$ of plausible views over the entire set X. This definition can be naturally extended to views over subsets of X. Specifically, let $Y \subseteq X$, let $x \in Y$, let $\mathcal{V}$ be a set of plausible views at the vertices of Y, and let $T = (Y, E_T, w_T)$ be a positively weighted tree. We say that $\mathcal{V}$ is a ***consistent set*** over subtree T with root x if there are bijections for each $y \in Y \setminus \{x\}$, $\ell_y : adj(y) \to \{1, 2, \ldots, \deg_y\}$, and a bijection $\ell_x : adj(x) \to \{1, 2, \ldots, \deg_x\} \setminus \{A_x^b\}$, such that:

(1) For each edge $e = (u, v) \in E_T$ and the corresponding views $V_u, V_v \in \mathcal{V}$, we have that V_u and V_v are consistent over $(\ell_u(e), \ell_v(e))$, and $w_T(e) = |A_u^d - A_v^d|$.

(2) For any view $V_u \in \mathcal{V}$, the following three conditions are equivalent (i) $A_u^b = null$, (ii) $A_u^d = 0$, and (iii) $u = x = \mathsf{a}$.

Comparing with our previous definition of consistency, observe that $\mathcal{V}$ is a consistent set over subtree T with root a, if and only if $\mathcal{V}$ is a consistent set over tree T.

Lemma 3.12. *Let X be an n-point metric space, with doubling dimension λ, and spread Δ. For any $\delta > 0$, $\epsilon > 0$, and $\mathsf{deg} > 0$ there is a*

$$n^2 \cdot \left(\frac{\Delta}{\epsilon}\right)^{\log_\delta(\mathsf{deg}/\epsilon)(O(\delta^2/\epsilon))^\lambda}$$

time algorithm to compute a $(1+\epsilon)\delta$ distortion embedding of X into a tree T of maximum degree deg if $\delta \geq \delta_{opt}(X, \mathsf{deg})$. If $\delta < \delta_{opt}(X, \mathsf{deg})$, this algorithm either computes an embedding of distortion $(1+\epsilon)\delta$ or (correctly) decides that $\delta < \delta_{opt}(X, \mathsf{deg})$.

Proof. First suppose that $\delta \geq \delta_{opt}(X, \mathsf{deg})$. Lemma 3.6 then guarantees the existence of a set of views, one for each $x \in X$, with parameters $(\mathsf{deg}, \sigma, c, (1+\sigma)\delta)$ that are consistent over some tree T. (Since if there is $(\mathsf{deg}, \sigma, c, (1+\sigma)\delta_{opt})$ set of consistent views then there is a $(\mathsf{deg}, \sigma, c, (1+\sigma)\delta)$ set of consistent views). Moreover, Lemma 3.11 implies that if there is a consistent set of views over some T with parameters $(\mathsf{deg}, \sigma, c, (1+\sigma)\delta)$ then T defines an embedding with distortion at most

$$\begin{aligned}\left(\left(1 + \frac{20}{c-1}\right) \Big/ \left(1 - \frac{14}{c-1}\right)\right) \cdot (1+\sigma) \cdot \delta &\leq \left(1 + \frac{20}{c-1}\right) \cdot \left(1 + \frac{28}{c-1}\right) \cdot (1+\sigma) \cdot \delta \\ &= \left(1 + \frac{20}{c-1} + \frac{28}{c-1} + \frac{560}{(c-1)^2}\right) \cdot (1+\sigma) \cdot \delta \leq \left(1 + \frac{608}{c-1}\right) \cdot (1+\sigma) \cdot \delta,\end{aligned}$$

which is at most $(1+\epsilon)\delta$ for $c = \frac{3 \times 608}{\epsilon} + 1$ and $\sigma = \epsilon/3$.[6] So set c and σ to these values and let $\delta' = (1+\sigma)\delta$. Then the above two statements combined imply that to prove the lemma, it suffices to give an algorithm which finds any consistent set of views (over some tree) with parameters $(\mathsf{deg}, \sigma, c, \delta')$, if one exists, and otherwise returns $\delta < \delta_{opt}(X, \mathsf{deg})$. We now describe a recursive algorithm which we then memoize to compute such a set of views.

Consider any collection $\mathcal{V}$ of views, with exactly one view V_x for each $x \in X$, with parameters $(\mathsf{deg}, \sigma, c, \delta')$. Given a weighted tree T on vertex set X, we first consider the simpler task of checking whether $\mathcal{V}$ is consistent over T. As discussed above, this is equivalent to saying that $\mathcal{V}$ is consistent over subtree T with root a. Let $T_\mathsf{a} = T$ and $\mathcal{V}_\mathsf{a} = \mathcal{V}$, and for any $x \neq \mathsf{a}$ in X, let T_x denote the subtree rooted at x and not containing a, and similarly let $\mathcal{V}_x$ be the subset of views over the vertices in this subtree. Also, let $adj'(x)$ denote all neighbors of x other than the one on the path

[6] Note that c can be made significantly smaller, however, in this paper to keep the calculations readable we are not optimizing constants.

to a, that is $adj'(x)$ are the neighbors of x in T_x (note $adj'(x) = adj(x)$ if $x = \mathsf{a}$). Observe that for any $x \in X$, $\mathcal{V}_x$ is a consistent set of views over subtree T_x with root x if and only if for every $y \in adj'(x)$ (1) $\mathcal{V}_y$ is a consistent set of views over subtree T_y with root y, and (2) V_x and V_y are consistent over $e = (x, y)$ with $w_{T_x}(x, y) = |A^d_x - A^d_y|$. Note that this is a recursive statement. Thus to check consistency of $\mathcal{V}_x$ over T_x, condition (1) can be checked by recursion, where the base case is when $adj'(x) = \emptyset$, and condition (2) can be checked by the algorithm of Lemma 3.5. To check if the full set $\mathcal{V}$ is consistent over T, we apply this recursive algorithm with $x = \mathsf{a}$.

This immediately implies a recursive algorithm for the harder problem of determining whether there exists any such collection of views $\mathcal{V}$ consistent over some tree T. Namely, consider all possible views with parameters $(\mathsf{deg}, \sigma, c, \delta')$ that are centered at the anchor a. For each such view V_{a}, we recursively determine if there is a collection of views containing V_{a}, which is consistent over a subtree T with root a. To do so consider all possible partitions of $X \setminus \{\mathsf{a}\}$ into deg_{a} subsets (i.e. subtrees). Then for each subset Z in a given partition we try all possible views over all members in Z as the root view, and for each such view V_x, if the view is consistent with V_{a} over the edge (a, x) (note the weight of the edge will then be $|A^d_{\mathsf{a}} - A^d_x|$), we then recursively check whether there is a collection of views over Z containing V_x, which is consistent over any subtree T_Z with root x. The correctness of this approach is apparent from the discussion above, however, the running time is exponential. Specifically, Lemma 3.2 bounds the number of possible views we must consider, but remembering the subsets and guessing how they are partitioned takes exponential time. However, we can now make use of Lemma 3.4, which states that for any view V_x, one can compute the unique partition $P = \{p_1, \ldots, p_{deg_x}\}$ of $X \setminus \{x\}$, such that if there is a feasible extension of V_x to an embedding defined by a tree T, then the sets in P must be the sets of vertices form each component of $T \setminus \{x\}$. Thus if x is a root with a view V_x over some subset Z then we can assume $Z = \cup_{i \in \{1,2,\ldots,\deg_x\} \setminus \{A^b_x\}} p_i$, and thus Z does not need to be passed as a parameter to the recursive problem. Moreover, rather than guessing all possible partitions of Z in this subproblem, we just use the partition $P \setminus \{p_{A^b_x}\}$.

Each subproblem of this recursive procedure is defined by a root $x \in X$ and a view V_x. Thus we can setup a dynamic programming table, index by (x, V_x) pairs, and then fill the table using the above recursive procedure and memoization.

For the running time, there are n choices for x, and $(1/\sigma) \cdot (c\Delta)^{O(\log_{\delta'}(c \cdot \mathsf{deg}))(2c(\delta')^2)^\lambda}$ choices for V_x by Lemma 3.2. Thus, the size of the table is

$$\frac{n}{\sigma} \cdot (c\Delta)^{O(\log_{\delta'}(c \cdot \mathsf{deg}))(2c(\delta')^2)^\lambda}.$$

Since we use memoization, each table entry is filled only once. For each table entry, we first compute its partition, and then independently for each (non-anchor) subset in the partition, and for each view V_z at a member z in the subset we check if V_x and V_z are consistent (which itself includes plausibility checks), and if so check the table entry (z, V_z). Ignoring the time spent in recursive calls, our algorithm thus spends at most

$$\begin{aligned} O(n \log_{\delta'}(c\Delta)) + \left(\mathsf{deg} \cdot \frac{n}{\sigma} \cdot (c\Delta)^{O(\log_{\delta'}(c \cdot \mathsf{deg}))(2c(\delta')^2)^\lambda}\right) \cdot O((2c(\delta')^2)^{2\lambda} \cdot \log_{\delta'}(c\Delta)) \\ = \frac{n}{\sigma} \cdot (c\Delta)^{O(\log_{\delta'}(c \cdot \mathsf{deg}))(2c(\delta')^2)^\lambda} \end{aligned}$$

time per table entry, where the first term is the time it takes to compute the partition (Lemma 3.4), and last part of the second term is the time to check for a pair of views whether each is plausible and whether they are consistent (Lemma 3.3 and Lemma 3.5). Therefore, the total running time of the algorithms is

$$\left(\frac{n}{\sigma} \cdot (c\Delta)^{O(\log_{\delta'}(c \cdot \mathsf{deg}))(2c(\delta')^2)^\lambda}\right) \cdot \left(\frac{n}{\sigma} \cdot (c\Delta)^{O(\log_{\delta'}(c \cdot \mathsf{deg}))(2c(\delta')^2)^\lambda}\right) = \frac{n^2}{\sigma^2} \cdot (c\Delta)^{O(\log_{\delta'}(c \cdot \mathsf{deg}))(2c(\delta')^2)^\lambda}$$

As discussed above, in order to obtain a $1+\epsilon$ approximation, we had set $c = O(\frac{1}{\epsilon})$, $\sigma = O(\epsilon)$, and $\delta' = (1+\sigma)\delta$, and thus the running time of our algorithms is

$$\frac{n^2}{\epsilon^2} \cdot \left(\frac{\Delta}{\epsilon}\right)^{\log_\delta(\mathsf{deg}/\epsilon)(O(\delta^2/\epsilon))^\lambda} = n^2 \cdot \left(\frac{\Delta}{\epsilon}\right)^{\log_\delta(\mathsf{deg}/\epsilon)(O(\delta^2/\epsilon))^\lambda}. \qquad \square$$

Theorem 3.13. *Let X be an n-point metric space, with doubling dimension λ and spread Δ. For any $\varepsilon > 0$ and $\mathsf{deg} > 1$, where $\delta_{opt} = \delta_{opt}(X, \mathsf{deg})$, there is an algorithm with running time*

$$n^2 \cdot \left(\frac{\Delta}{\varepsilon}\right)^{\log(\mathsf{deg}/\varepsilon)\cdot(1/\varepsilon)^{\lambda+1}\cdot(O(\delta_{opt}))^{2\lambda}}$$

to compute a $(1+\varepsilon)\delta_{opt}$ distortion embedding of X into a tree T of maximum degree deg.

Proof. Consider the set $L = \{\delta_i = (1+\varepsilon/2)^i | 1 \le i \le n\Delta\}$. Our algorithm calls the procedure of Lemma 3.12 with $\epsilon = \varepsilon/3$ and $\delta = \delta_i$, in increasing order of δ_i, until it first successfully finds an embedding. Note that since X can always be embedded into a path with distortion at most $n\Delta$, our algorithm will always find an embedding. To bound the distortion of the computed embedding, let $1 \le j \le n\Delta$ be such that $(1+\varepsilon/2)^{j-1} \le \delta_{opt}(X, \mathsf{deg}) \le (1+\varepsilon/2)^j$. Then the procedure of Lemma 3.12 will return an embedding of distortion at most $(1+\varepsilon/2)^j \cdot (1+\varepsilon/3)$ if it is called with parameters $\delta = \delta_j = (1+\varepsilon/2)^j$ and $\epsilon = \varepsilon/3$. Thus we get an embedding with distortion at most

$$(1+\varepsilon/2)^j \cdot (1+\varepsilon/3) \le \delta_{opt} \cdot (1+\varepsilon/2) \cdot (1+\varepsilon/3) \le \delta_{opt} \cdot (1+\varepsilon).$$

It remains to bound the running time of our algorithm. We call the procedure of Lemma 3.12 $O(\log_{1+\varepsilon/2}(\delta_{opt}))$ times. The running time of each of these procedure calls is bounded by

$$n^2 \cdot \left(\frac{\Delta}{\epsilon}\right)^{\log_\delta(\mathsf{deg}/\epsilon)(O(\delta^2/\epsilon))^\lambda} = n^2 \cdot \left(\frac{\Delta}{\varepsilon}\right)^{\log_{1+\varepsilon/2}(3\cdot\mathsf{deg}/\varepsilon)(O(\delta^2/\varepsilon))^\lambda}. \qquad (8)$$

We know via Taylor series expansion, that for $0 \le x \le 1$, $\log(1+x) \ge x - x^2/2 = (x/2)(2-x) \ge x/2$. Therefore since $0 < \varepsilon \le 1$,

$$\log_{1+\varepsilon/2}(3\cdot\mathsf{deg}/\varepsilon) = \frac{\log(3\cdot\mathsf{deg}/\varepsilon)}{\log(1+\varepsilon/2)} \le \frac{\log(3\cdot\mathsf{deg}/\varepsilon)}{\varepsilon/2} = O\left(\frac{\log(\mathsf{deg}/\varepsilon)}{\varepsilon}\right)$$

Substituting in (8) we find out that the running time of each call is bounded by

$$n^2 \cdot \left(\frac{\Delta}{\varepsilon}\right)^{\log_{1+\varepsilon/2}(3\cdot\mathsf{deg}/\varepsilon)(O(\delta^2/\varepsilon))^\lambda} = n^2 \cdot \left(\frac{\Delta}{\varepsilon}\right)^{\log(\mathsf{deg}/\varepsilon)\cdot(1/\varepsilon)\cdot(O(\delta^2/\varepsilon))^\lambda}$$

Thus the total running time is

$$O(\log_{1+\varepsilon/2}(\delta_{opt})) \cdot n^2 \cdot \left(\frac{\Delta}{\varepsilon}\right)^{\log(\mathsf{deg}/\varepsilon)\cdot(1/\varepsilon)\cdot(O(\delta^2/\varepsilon))^\lambda} = n^2 \cdot \left(\frac{\Delta}{\varepsilon}\right)^{\log(\mathsf{deg}/\varepsilon)\cdot(1/\varepsilon)^{\lambda+1}\cdot(O(\delta_{opt}))^{2\lambda}} \qquad \square$$

Corollary 4.3 in the next section shows that any tree with doubling dimensions λ can be embedded with $(1+\varepsilon)$ distortion into a tree with maximum degree $(O(1/\varepsilon))^\lambda$. Thus the above lemma statement can be strengthened to remove the degree assumption, yielding Theorem 1.1.

Proof of Theorem 1.1. By Corollary 4.3, for any $\epsilon > 0$ there is a tree $T = (X, E_T, w_T)$ that defines an embedding of distortion at most $(1+\epsilon)\delta_{opt}(X)$ and that has maximum degree $O((1/\epsilon)^\lambda)$. Therefore, Theorem 3.13 gives a $(1+\varepsilon)$ approximation algorithm by setting deg to $O((1/\epsilon)^\lambda)$ and $\epsilon = \varepsilon/3$. The running time of the algorithm is

$$n^2 \cdot \left(\frac{\Delta}{\varepsilon}\right)^{\lambda\log(1/\varepsilon)\cdot(1/\varepsilon)^{\lambda+1}\cdot(O(\delta_{opt}))^{2\lambda}} = n^2 \cdot \left(\frac{\Delta}{\varepsilon}\right)^{\log(1/\varepsilon)\cdot(1/\varepsilon)^{\lambda+1}\cdot(O(\delta_{opt}))^{2\lambda}} \qquad \square$$

4 Bounded degree trees as host metrics

In this section, we show that a doubling metric space has a nearly optimal bounded degree tree spanner. In particular, this result implies that a doubling tree can be embedded into a bounded degree tree with same vertex set nearly isometrically. These results imply that considering bounded degree trees is sufficient when we study embedding into trees or computing geometric tree spanners. Specifically, they are used in the proofs of Theorem 1.1 and Theorem 1.3. We start by describing a simple transformation.

Let (X, d_X) be a metric space. Given a subset $V \subseteq X$, and a point $v \in V$, define the *star* of v with respect to the subset V, to be the star graph $H = (V, E_H, w_H)$ with center v such that $E_H = \{(v, x) \mid x \in V \setminus \{v\}\}$ and $w_H(v, x) = d_X(v, x)$ for any $(v, x) \in E_H$. Let $\alpha > 1$ be some fixed value. For any non-negative integer i, let $L_i = \{u \in V | \alpha^i \leq d_X(v, u) < \alpha^{i+1}\}$. We refer to L_i's as ***layers***, and let $L_{odd} = \bigcup_{(i \text{ is odd})} L_i$, and $L_{even} = \bigcup_{(i \text{ is even})} L_i$.

We now define a partition $\{P_i\}_{1 \leq i \leq p}$ of the set L_{odd} using the following iterative procedure. For any round j, if $L_{odd} \setminus (\bigcup_{1 \leq i < j} P_i) = \emptyset$ then the procedure terminates, otherwise define P_j to be any maximal subset of vertices in $L_{odd} \setminus (\bigcup_{1 \leq i < j} P_i)$ that has at most one vertex in each layer. Using the same iterative procedure we define the analogous partition $\{Q_i\}_{1 \leq i \leq q}$ of the set L_{even}.

Let a *spider* with center v refer to any connected graph, where all vertices other than v have degree ≤ 2. The ***star to spider transformation*** of the star H into the spider $H' = (V, E_{H'}, w_{H'})$ is then defined by specifying edges and weights of H' as follows (see Figure 3). For each $1 \leq i \leq p$, add a path π_i that visits every vertex $u \in P_i \cup \{v\}$ in increasing order of the $d_X(v, u)$ distances. Similarly, for each $1 \leq i \leq q$, add a path γ_i that visits every vertex $w \in Q_i \cup \{v\}$ in increasing order of the $d_X(v, w)$ distances. Finally, for each edge $(x, y) \in E_{H'}$, set $w_{H'}(x, y) = d_X(x, y)$.

Figure 3: A star to spider transformation. Blue vertices belong to odd layers, and purple vertices belong to even layers.

Lemma 4.1. *Let (X, d_X) be a metric space, let $H = (V, E_H, w_H)$ be a star with center $v \in V$, and let $H' = (V, E_{H'}, w_{H'})$ be the graph obtained from H via the star to spider transformation, where w_H and $w_{H'}$ are restrictions of d_X into E_H and $E_{H'}$, respectively.*

(1) H' is a spider with center v of degree at most $2 \cdot \max(|L_i|)$.

(2) For each $(v, x) \in E_H$, we have that $d_{H'}(v, x) \leq \left(1 + \frac{2}{\alpha - 1}\right) w_H(v, x)$.

Proof. (1) $\{L_{odd}, L_{even}\}$ is a partition of V, and $\{P_1, P_2, \ldots, P_p\}$ and $\{Q_1, Q_2, \ldots, Q_q\}$ are partitions of L_{odd} and L_{even}, respectively. Therefore, $\{P_1, P_2, \ldots, P_p, Q_1, Q_2, \ldots, Q_q\}$ is a partition of V. It follows that all π_i's and γ_i's are vertex disjoint paths except for the common vertex v. So, H' is a spider with center v, and $p + q$ legs. By the maximality of the P_i's and Q_i's, $p = \max_{(i \text{ is odd})} |L_i|$, and $q = \max_{(i \text{ is even})} |L_i|$, therefore, $p + q \leq 2 \max_i |L_i|$.

(2) Let $(v, x) \in E_H$. Suppose $x \in L_{odd}$, the proof for the other case is similar. Let π_i be the leg of

H' that contains x, where $\pi_i[x,v] = (x = u_k, \ldots, u_1 = v)$. We have:

$$\begin{aligned} d_{H'}(x,v) &= len_{H'}(\pi_i[x,v]) = \sum_{i=2}^{k} w_{H'}(u_i,u_{i-1}) = \sum_{i=2}^{k} d_X(u_i,u_{i-1}) \\ &\leq \sum_{i=2}^{k} (d_X(u_i,v) + d_X(v,u_{i-1})) = \sum_{i=2}^{k} (w_H(u_i,v) + w_H(v,u_{i-1})) \\ &= w_H(u_k,v) + w_H(u_1,v) + 2\sum_{i=2}^{k-1} w_H(u_i,v) = w_H(x,v) + 2\sum_{i=2}^{k-1} w_H(u_i,v). \end{aligned} \quad (9)$$

On the other hand, for each $2 \leq i \leq k-1$, we have that $w_H(u_i,v) \leq w_H(u_{i+1},v)/\alpha$, since u_i and u_{i+1} belong to two different non-consecutive layers. It follows that, for all $2 \leq i \leq k-1$,

$$w_H(u_i,v) \leq \alpha^{i-k} w_H(u_k,v) = \alpha^{i-k} w_H(x,v).$$

Substituting in (9) we obtain

$$\begin{aligned} d_{H'}(x,v) &\leq w_H(x,v) + 2\sum_{i=2}^{k-1} \alpha^{i-k} w_H(x,v) \leq w_H(x,v)\left(1 + 2\sum_{j=1}^{k-2} \alpha^{-j}\right) \\ &\leq w_H(x,v)\left(1 + \frac{2\alpha^{-1}}{1-\alpha^{-1}}\right) \leq w_H(x,v)\left(1 + \frac{2/\alpha}{(\alpha-1)/\alpha}\right) \leq w_H(x,v)\left(1 + \frac{2}{\alpha-1}\right). \quad \square \end{aligned}$$

Given any metric space, the following lemma guarantees the existence of a low-degree spanning tree, whose distortion is very close to the optimal distortion of any spanning tree. For a metric $d_X : X \times X \to \mathbb{R}^+$ and a set of pairs $E \subseteq X \times X$, in the following $d_X[E] : E \to \mathbb{R}$ denotes the ***restriction*** of d_X to E, that is for any $(x,y) \in E$, $d_X[E](x,y) = d_X(x,y)$.

Lemma 4.2. *Let (X,d_X) be a metric space of doubling dimension λ. Let $T = (X,E,w)$ be a tree with $w = d_X[E]$. Suppose the identity map from (X,d_X) to (X,d_T) has distortion δ. For any $\varepsilon > 0$, there is a tree $T' = (X,E',w')$ with maximum degree $(O(\delta/\varepsilon))^\lambda$ such that $w' = d_X[E']$ and the identity map from (X,d_X) to $(X,d_{T'})$ has distortion at most $(1+\varepsilon)\delta$.*

Proof. Pick an arbitrary vertex $r \in X$ to be the root of T. For each $v \in X$, let H_v be the star with center v, whose leaves are the children of v in T. The set $\mathcal{H} = \{H_v | v \in V\}$ is a set of edge disjoint stars whose edge sets partition E.

To obtain T', for every $v \in V$, we replace the star H_v with a spider H'_v obtained via the star to spider transformation. Any two vertices of any star remain connected after the transformation, thus T' is connected. Additionally, the star to spider transformation preserves the number of edges, therefore T' is a tree.

Let I_{XT} be the (X,d_X) to (X,d_T) identity map, let $I_{XT'}$ be the (X,d_X) to $(X,d_{T'})$ identity map, and let $I_{TT'}$ be the (X,d_T) to $(X,d_{T'})$ identity map. We bound the distortion of $I_{XT'}$. First, by the definition of T', the map $I_{XT'}$ has contraction one. To bound its expansion, note $I_{XT'} = I_{XT} \circ I_{TT'}$. Since I_{XT} has contraction one, its expansion is δ. Thus, it remains to show that the expansion of $I_{TT'}$ is bounded. By Lemma 2.2, it suffices to bound the expansion of every edge of T by $(1+\varepsilon)$. Let $(x,y) \in E$. Without loss of generality assume y is the parent of x, thus $(x,y) \in H_y$. By, Lemma 4.1, the x-to-y path in H'_y has length at most $(1 + \frac{2}{\alpha-1})w(x,y)$. Thus to obtain the $(1+\varepsilon)$ upper bound we only need to set

$$\frac{2}{\alpha-1} = \varepsilon \Rightarrow \alpha = \frac{2}{\varepsilon} + 1.$$

Next, we bound the degree of each vertex v in T', by bounding the number of vertices in each layer of H_v (using Lemma 4.1-(1)). Let x and y be any two vertices at layer i of H_v, that is $\alpha^i \le d_X(v,x) < \alpha^{i+1}$, and $\alpha^i \le d_X(v,y) < \alpha^{i+1}$. In particular, $d_T(x,y) \ge 2\alpha^i$, and since I_{XT} has expansion at most δ, $d_X(x,y) \ge 2\alpha^i/\delta$. Therefore, by Lemma 2.1, the number of points in layer i is at most,

$$\left(\frac{2\alpha^{i+1}}{2\alpha^i/\delta}\right)^{\lambda} = (\alpha\delta)^{\lambda}.$$

Overall, each vertex is the center of at most one star and the leaf of at most one star. After the transformation the center of each spider has degree at most $2\max(|L_i|) \le 2(\alpha\delta)^{\lambda}$, and each non-center has degree at most 2. Hence, the degree of any vertex of T' is at most

$$2(\alpha\delta)^{\lambda} + 2 = 2(2\delta/\varepsilon + \delta)^{\lambda} + 2 = (O(\delta/\varepsilon))^{\lambda}. \qquad \square$$

For a tree $T = (V, E, w)$, by setting $X = V$ and $d_X = d_T$ we obtain the following useful corollary.

Corollary 4.3. *Let $T = (V, E, w)$ be a tree of doubling dimension λ. For any $\varepsilon > 0$, there is a tree $T' = (X, E', w')$ with maximum degree $(O(1/\varepsilon))^{\lambda}$ such that the identity map from (V, d_T) to $(V, d_{T'})$ has distortion at most $(1+\varepsilon)$.*

5 Tree spanners

In this section, we consider the closely related problem of finding tree spanners with minimum stretch. Lets start by considering a generalization of both the low-distortion trees and low-stretch tree spanners problems.

Let (X, d_X) be a finite metric space. Similar to the low-distortion embedding problem, we would like to embed X into a tree $T = (X, E_T, w_T)$. However, we have a set of constraints on possible edges and edge weights of T. Specifically, we have a ***constraint function*** $h : X \times X \to 2^{\mathbb{R}^+}$, which specifies the set of ***permitted weights*** for every pair of vertices $x, y \in X$ if we choose to include (x, y) in E_T. In particular, an edge (x, y) is banned if $h(x, y) = \emptyset$. A tree is a ***permitted tree*** if all its edge weights are permitted. Given (X, d_X) and h, the ***constrained embedding problem*** asks for the minimum distortion embedding of X into any permitted tree. Note that the minimum stretch tree spanner problem for a graph $G = (X, E_G, w_G)$ is equivalent to the constrained embedding problem for (X, d_G) and h, where $h(x, y) = \{w_G(x, y)\}$ if $(x, y) \in E_G$, and $h(x, y) = \emptyset$, otherwise. Moreover, the minimum distortion embedding problem is equivalent to the constrained embedding problem for (X, d_X) and h, where $h(x, y) = \mathbb{R}^+$ for all $x, y \in X$.

We modify the algorithm of Lemma 3.12 to solve the constrained embedding problem. Let $\boldsymbol{\delta_{opt}(X, \mathsf{deg}, h)}$ denote the minimum distortion of any embedding of (X, d_X) into a tree with vertex set X, maximum degree at most deg, and which is permitted with respect to h.

One issue is that our algorithm works with a discrete step size σ, while h might allow edge weights that are not integer multiples of σ. To resolve this issue, in a preprocessing step we change h to h', where $h'(x, y) = \{\lceil a/\sigma \rceil \cdot \sigma \mid a \in h(x, y)\}$. The proof of Lemma 3.1 implies that $\delta_{opt}(X, \mathsf{deg}, h') \le (1+\sigma)\delta_{opt}(X, \mathsf{deg}, h)$. Our algorithm then solves the problem for X, deg, and h' to find a tree $T' = (X, E_{T'}, w_{T'})$ that is permitted with respect to h'. To obtain a permitted tree with respect to h in a postprocessing step we modify $w_{T'}$ as follows. For each $(x, y) \in E_{T'}$, we set $w_T(x, y)$ to the largest permitted value (with respect to h) that is at most $w_{T'}(x, y)$. Let δ' and δ be distortions of identity maps from (X, d_X) to $(X, d_{T'})$ and from (X, d_X) to (X, d_T), respectively. Applying Lemma 3.1 in the reverse direction implies that $\delta \le (1+\sigma) \cdot \delta'$. Hence, the preprocessing and postprocessing introduce a factor of at most $(1+\sigma)^2$ in the final distortion.

Now, let $\sigma > 0$, and let h' be the refined constraint function with step size σ. Also, let $\delta'_{opt} = \delta_{opt}(X, \mathsf{deg}, h')$. A modification of the argument of Lemma 3.6 shows the existence of a consistent set of views with parameters $(\mathsf{deg}, \sigma, c, \delta'_{opt})$ over a permitted tree T'. Given any consistent set of views over any tree Lemma 3.11 guarantees distortion at most $(1-14/(c-1))^{-1} \cdot (1+20/(c-1))\delta'_{opt}$ for the identity map to that tree.

We slightly modify the dynamic programming of Lemma 3.12 to compute a permitted tree with a consistent set of views over it. Specifically, whenever we check consistency between two views V_x and V_y over an edge (x, y), we make sure that $|A^d_x - A^d_y| \in h(x, y)$, that is $|A^d_x - A^d_y|$ is a permitted weight for (x, y). The rest of the algorithm remains intact. Together with the preprocessing and the postprcessing step, we obtain an algorithm that is guaranteed to return a tree with distortion at most

$$\left(1 - \frac{14}{c-1}\right)^{-1} \cdot \left(1 + \frac{20}{c-1}\right) \cdot \delta'_{opt} \leq \left(1 + \frac{608}{c-1}\right) \cdot (1+\sigma)^2 \cdot \delta_{opt}.$$

So by setting $c = 7 \times 608/\epsilon + 1$ and $\sigma = \epsilon/7$, we are guaranteed that the distortion of our output is at most $(1+\epsilon)\delta_{opt}$, and the following theorem follows.

Lemma 5.1. *Let X be an n-point metric space, with doubling dimension λ, and spread Δ. Also, let $h : X \times X \to 2^{\mathbb{R}^+}$ specify permitted edge weights. For any $\delta > 0$, $\epsilon > 0$, and $\mathsf{deg} > 0$ there is a*

$$n^2 \cdot \Delta^{\log_\delta(\mathsf{deg}/\epsilon)(O(\delta^2/\epsilon))^\lambda}$$

time algorithm to compute a $(1+\epsilon)\delta$ distortion embedding of X into a permitted tree T (with respect to h) of maximum degree deg if $\delta \geq \delta_{opt}(X, \mathsf{deg}, h)$. If $\delta < \delta_{opt}(X, \mathsf{deg}, h)$, this algorithm either computes an embedding of distortion $(1+\epsilon)\delta$ or (correctly) decides that $\delta < \delta_{opt}(X, \mathsf{deg}, h)$.

The general version of Theorem 3.13 follows, by the exact same proof.

Theorem 5.2. *Let X be an n-point metric space, with doubling dimension λ and spread Δ. Also, let $h : X \times X \to 2^{\mathbb{R}^+}$ specify permitted edge weights. For any $\varepsilon > 0$ and $\mathsf{deg} > 1$, where $\delta_{opt} = \delta_{opt}(X, \mathsf{deg}, h)$, there is a*

$$n^2 \cdot \left(\frac{\Delta}{\varepsilon}\right)^{\log(\mathsf{deg}/\varepsilon)\cdot(1/\varepsilon)^{\lambda+1}\cdot(O(\delta_{opt}))^{2\lambda}}$$

time algorithm to compute a $(1+\varepsilon)\delta_{opt}$ distortion embedding of X into a permitted tree T of maximum degree deg.

Now, we are ready to prove our spanner results using Thorem 5.2.

Proof of Theorem 1.3. Let $\delta_{opt} = str(X)$ be the minimum possible stretch of any spanning tree of X, and let $str(X, \mathsf{deg})$ denote the minimum possible stretch of any spanning tree of X with maximum degree at most deg. Let $h : X \times X \to 2^{\mathbb{R}^+}$ be defined as follows. For each $x, y \in X$, set $h(x, y) = \{d_X(x, y)\}$. Note that $\delta_{opt}(X, \mathsf{deg}, h) = str(X, \mathsf{deg})$ and let δ denote this value. The algorithm of Theorem 5.2 can find a permitted tree with respect to h and an embedding of distortion at most $(1+\varepsilon/3)\cdot\delta$ in time

$$n^2 \cdot \left(\frac{\Delta}{\varepsilon}\right)^{\log(\mathsf{deg}/\varepsilon)\cdot(1/\varepsilon)^{\lambda+1}\cdot(O(\delta))^{2\lambda}},$$

for any $\varepsilon/3$ and deg. On the other hand, Lemma 4.2 implies that there exists $\deg = (O(1/\varepsilon))^\lambda$ such that $\delta = str(X, \deg) \leq (1+\varepsilon/3)\cdot str(X) = \delta_{opt}$. Thus, a $(1+\varepsilon)$ approximation of the minimum stretch tree can be computed in time

$$n^2 \cdot \left(\frac{\Delta}{\varepsilon}\right)^{\log(1/\varepsilon)\cdot(1/\varepsilon)^{\lambda+1}\cdot(O(\delta_{opt}))^{2\lambda}}. \qquad \square$$

Proof of Theorem 1.4. Let $h : X \times X \to 2^{\mathbb{R}^+}$ be defined as follows. For each $x, y \in X$, set $h(x, y) = \{w(x, y)\}$ if $(x, y) \in E$, and set $h(x, y) = \emptyset$ otherwise. For any $\varepsilon > 0$, Theorem 5.2 finds a permitted embedding into a tree T of distortion at most $(1+\varepsilon)\delta_{opt}((X, d_G), \mathsf{deg}, h)$. The constraint function ensures that T is a spanning tree of G. Also, as h allows all spanning trees of G and no other tree, we have $\delta_{opt}((X, d_G), \mathsf{deg}, h)$ is equal to the minimum stretch of all spanning trees. □

References

[ABF+99] Richa Agarwala, Vineet Bafna, Martin Farach, Mike Paterson, and Mikkel Thorup. On the approximability of numerical taxonomy (fitting distances by tree metrics). *SIAM J. Comput.*, 28(3):1073–1085, February 1999.

[BCIS05] Mihai Badoiu, Julia Chuzhoy, Piotr Indyk, and Anastasios Sidiropoulos. Low-distortion embeddings of general metrics into the line. In *Proceedings of the 37th Annual ACM Symposium on Theory of Computing*, STOC '05, pages 225–233. ACM, 2005.

[BDH+08] Mihai Bădoiu, Erik D. Demaine, Mohammadtaghi Hajiaghayi, Anastasios Sidiropoulos, and Morteza Zadimoghaddam. Ordinal embedding: Approximation algorithms and dimensionality reduction. In *Proceedings of the 11th International Workshop, APPROX 2008, and 12th International Workshop, RANDOM 2008 on Approximation, Randomization and Combinatorial Optimization: Algorithms and Techniques*, APPROX '08 / RANDOM '08, pages 21–34, Berlin, Heidelberg, 2008. Springer-Verlag.

[BIS07] Mihai Badoiu, Piotr Indyk, and Anastasios Sidiropoulos. Approximation algorithms for embedding general metrics into trees. In *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '07, pages 512–521, Philadelphia, PA, USA, 2007. Society for Industrial and Applied Mathematics.

[Bun71] Peter Buneman. The Recovery of Trees from Measures of Dissimilarity. In D. G. Kendall and P. Tautu, editors, *Mathematics the the Archeological and Historical Sciences*, pages 387–395. Edinburgh University Press, 1971.

[CC95] Leizhen Cai and Derek G. Corneil. Tree spanners. *SIAM J. Discret. Math.*, 8(3):359–387, August 1995.

[CDN+10] Victor Chepoi, Feodor F. Dragan, Ilan Newman, Yuri Rabinovich, and Yann Vaxes. Constant approximation algorithms for embedding graph metrics into trees and outerplanar graphs. In Maria J. Serna, Ronen Shaltiel, Klaus Jansen, and Jose D. P. Rolim, editors, *APPROX-RANDOM*, volume 6302 of *Lecture Notes in Computer Science*, pages 95–109. Springer, 2010.

[CHL07] Otfried Cheong, Herman Haverkort, and Mira Lee. Computing a minimum-dilation spanning tree is NP-hard. In *Proceedings of the Thirteenth Australasian Symposium on Theory of Computing - Volume 65*, CATS '07, pages 15–24, Darlinghurst, Australia, Australia, 2007. Australian Computer Society, Inc.

[DH98] Michael J. Demmer and Maurice Herlihy. The arrow distributed directory protocol. In *Proceedings of the 12th International Symposium on Distributed Computing*, DISC '98, pages 119–133, London, UK, UK, 1998. Springer-Verlag.

[EP08] Yuval Emek and David Peleg. Approximating minimum max-stretch spanning trees on unweighted graphs. volume 38, pages 1761–1781, Philadelphia, PA, USA, December 2008. Society for Industrial and Applied Mathematics.

[Epp00] David Eppstein. Spanning trees and spanners. In Jörg-Rudiger Sack and Jorge Urrutia, editors, *Handbook of Computational Geometry*, chapter 9, pages 425–461. Elsevier, 2000.

[EW05] David Eppstein and Kevin A. Wortman. Minimum dilation stars. In *Proceedings of the Twenty-first Annual Symposium on Computational Geometry*, SCG '05, pages 321–326, New York, NY, USA, 2005. ACM.

[FFL+13] Michael Fellows, Fedor V. Fomin, Daniel Lokshtanov, Elena Losievskaja, Frances Rosamond, and Saket Saurabh. Distortion is fixed parameter tractable. *ACM Trans. Comput. Theory*, 5(4):16:1–16:20, November 2013.

[FGvL11] Fedor V. Fomin, Petr A. Golovach, and Erik Jan van Leeuwen. Spanners of bounded degree graphs. *Inf. Process. Lett.*, 111(3):142–144, January 2011.

[GKL03] Anupam Gupta, Robert Krauthgamer, and James R. Lee. Bounded geometries, fractals, and low-distortion embeddings. In *44th Symposium on Foundations of Computer Science (FOCS 2003), 11-14 October 2003, Cambridge, MA, USA, Proceedings*, pages 534–543, 2003.

[Gup01] Anupam Gupta. Steiner points in tree metrics don't (really) help. In *Proceedings of the Twelfth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '01, pages 220–227, Philadelphia, PA, USA, 2001. Society for Industrial and Applied Mathematics.

[HIL98] Johan Håstad, Lars Ivansson, and Jens Lagergren. Fitting points on the real line and its application to RH mapping. In *Proceedings of the 6th Annual European Symposium on Algorithms*, ESA '98, pages 465–476, London, UK, UK, 1998. Springer-Verlag.

[IM04] Piotr Indyk and Jiří Matoušek. Low-distortion embeddings of finite metric spaces. In *Handbook of Discrete and Computational Geometry*, pages 177–196. CRC Press, 2004.

[Ind01] Piotr Indyk. Algorithmic applications of low-distortion geometric embeddings. In *Proceedings of the 42Nd IEEE Symposium on Foundations of Computer Science*, FOCS '01, pages 10–, Washington, DC, USA, 2001. IEEE Computer Society.

[KRS04] Claire Kenyon, Yuval Rabani, and Alistair Sinclair. Low distortion maps between point sets. In *Proceedings of the Thirty-sixth Annual ACM Symposium on Theory of Computing*, STOC '04, pages 272–280, New York, NY, USA, 2004. ACM.

[KW99] Junhyong Kim and Tandy Warnow. Tutorial on phylogenetic tree estimation. In *Intelligent Systems for Molecular Biology*. Press, 1999.

[LW08] Christian Liebchen and Gregor Wünsch. The zoo of tree spanner problems. *Discrete Appl. Math.*, 156(5):569–587, March 2008.

[Mat13] Jiří Matoušek. *Lecture notes on metric embeddings*, 2013.
Available at: `http://kam.mff.cuni.cz/~matousek/ba-a4.pdf`.

[NR15] Amir Nayyeri and Benjamin Raichel. Reality distortion: Exact and approximate algorithms for embedding into the line. In *IEEE 56th Annual Symposium on Foundations of Computer Science, FOCS 2015, Berkeley, CA, USA, 17-20 October, 2015*, pages 729–747, 2015.

[NR17] Amir Nayyeri and Benjamin Raichel. A treehouse with custom windows: Minimum distortion embeddings into bounded treewidth graphs. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2017, Barcelona, Spain, Hotel Porta Fira, January 16-19*, pages 724–736, 2017.

[Pap15] Ioannis Papoutsakis. Tree spanners of bounded degree graphs. *CoRR*, abs/1503.06822, 2015.

[PR01] David Peleg and Eilon Reshef. Low complexity variants of the arrow distributed directory. *J. Comput. Syst. Sci.*, 63(3):474–485, November 2001.