# Approximation Algorithms for Metric Embedding Problems

Kedar Dhamdhere

CMU-CS-05-152

June 2005

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

**Thesis Committee:**
R. Ravi, Chair
Anupam Gupta
Guy Blelloch
Piotr Indyk, MIT

*Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy.*



This research was sponsored by the National Science Foundation (NSF) under grant nos. NSF CCF 04-30751, ITR grant CCR-0122581 (The ALADDIN Project) and NSF CCR-0105548.

**Keywords:** Approximation Algorithms, Metric Embedding, Distortion, Line metric, Spanning Trees

## Abstract

We initiate the study of metric embedding problems from an approximation point of view. Metric embedding is a map from a guest metric to a host metric. The quality of the embedding is defined in terms of distortion, the ratio by which pairwise distances get skewed in the host metric. While metric embeddings in general have received quite a lot of attention in theory community, most of the results about distortion prove uniform bounds that work for various families of host and guest metric.

In this dissertation, we address the question: how to find the best embedding of the particular input metric into a host metric. We consider the real line as the host metric in our study. We consider the following measures of quality of an embedding: distortion, average distortion and additive distortion. The distortion is the maximum ratio by which a pairwise distance gets stretched in a non-contracting embedding. We give $O(\sqrt{n})$-approximation for the distortion of embedding an unweighted graph metric to a line metric. The average distortion is the ratio of average distance in the embedded metric to that in the input metric. We give a 17-approximation for the average distortion when embedding an arbitrary finite metric to a line metric. The additive distortion is the total absolute difference between input and output distances. We provide an $O(\sqrt{\log n})$-approximation for this objective function. We also show NP-hardness of these problems.

We also consider the problem of linear ordering of a metric, i.e. assigning numbers from 1 through n to the points in the metric, so as to minimize the 'stretch'. The stretch is the maximum pairwise distance in the ordering divided by the distance in the input metric. For this problem, we give $O(\log^3 n)$ approximation.

Finally, we consider the problem of constructing a probabilistic embedding of a graph into its spanning trees. We give a simple $O(\log^2 n)$-approximation algorithm that improves on the algorithm of Elkin et al. Elkin et al. [2005].

# Acknowledgements

First of all, I would like to thank my advisor R. Ravi for his invaluable guidance, support and insights. I would also like to thank my co-advisor Guy Blelloch.

I consider myself fortunate to have Anupam Gupta in the Department. I will be indebted to him for helping me grow as a researcher.

I would also like to thank Piotr Indyk for generously offering help with my thesis.

I want to thank all the theory folks at CMU for providing such a wonderful and stimulating research environment. Special thanks to Russell Schwartz, Avrim Blum, Harald Räcke and Chris Olston. I have learnt a lot from each of you.

Thanks to Nikhil Bansal, Amitabh Sinha, Jochen Könemann, Ke Yang, Shuchi Chawla, Srinath Sridhar, Amit Manjhi and Sandeep Pandey for being such great friends, colleagues and collaborators.

Finally, I would like to thank my family for their unconditional love and encouragement and instilling in me the love of learning. A special thanks to Anupriya for all her love and understanding.

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

Over the past decade, metric embeddings have been objects of much attention in theoretical computer science. This has been largely due to their many algorithmic applications, which range from simplifying the structure of the input data for approximation and online problems (Arora [1996, 1998], Bartal [1996], Bartal et al. [1997], Fakcharoenphol et al. [2003b], Garg et al. [2000], Kleinberg and Tardos [2002]), serving as convenient relaxations of important NP-hard problems Aumann and Rabani [1998], Blum et al. [2000], Bourgain [1985], Călinescu et al. [2001], Feige [2000], Linial et al. [1995] or simply by being the object of study ([Agarwala et al., 1999, Farach et al., 1995]) arising from applications such as computational biology. Embedding techniques have become an indispensable addition to the algorithm designer's toolbox, providing powerful and elegant solutions to many algorithmic problems (see, e.g., [Matoušek, 2002, Chapter 15] and [Indyk, 2001] for surveys).

An embedding of a metric $(V, d)$ into *host* metric $(H, \delta)$ is a map $f : V \to H$. The quality of this map is measured by how closely the distances between points in $d$ closely resemble those between their images in $\delta$. An embedding $f$ is called *non-contracting* if the map $f$ does not decrease any of the distances, i.e., $d(x, y) \leq \delta(f(x), f(y))$ for all $x, y \in V$. (In the sequel, we will abbreviate $\delta(f(x), f(y))$ to $\delta(x, y)$.) An important measure of the quality of a non-contracting embedding $f$ is the *distortion* $D = D(f)$, which is:

$$\text{distortion } D = \max_{x,y \in V} \frac{\delta(x, y)}{d(x, y)}. \tag{1.1}$$

(We note that a more general definition of distortion can be given that is scale-free; hence the restriction of non-contracting embeddings used here is without loss of generality.)

While many embedding techniques and algorithms are known, the analyses for these embeddings usually only offer uniform bounds on the distortion of the embeddings; few results which address the problem of minimizing the distortion required for embedding a *given* metric into the host space. In fact, very few results show how to even *approximate* the distortion to better than these uniform bounds.

This is perhaps best shown by a concrete example: [Matoušek, 1990] proved that *any* metric $(V, d)$ can be embedded into the real line with distortion $O(|V|)$; furthermore, the result is existentially tight, as the $n$-cycle cannot be embedded into the line with distortion $o(|V|)$ (see, e.g., Rabinovich and Raz [1998], Gupta [2001]). However, no algorithm is known for this problem which offers *per-instance* guarantees; even if a metric $(X, d)$ may be embeddable into $\mathbb{R}$ with distortion $D = O(1)$, the known algorithms do not seem to guarantee that the embedding they output has distortion, say, that is within $O(|V|^{1-\epsilon})$ times $D$.

This is the case with most problems in embeddings: while uniform upper bounds are known for embeddings of many different families of metrics (e.g., general metrics, planar graph metrics, tree metrics) into a variety of host spaces (e.g., the Minkowski $\ell_p$ spaces, distributions of trees), very little is known about how to approximate the optimal distortion given a *fixed* metric $(V, d)$ and a host space. One notable exception is the remark of [Linial et al., 1995] that the optimal embedding of any finite metric into (unbounded dimensional) Euclidean spaces to minimize distortion can be computed as a solution to a semi-definite program. Another one is the result by [Kenyon et al., 2004].

In this work we focus on studying the metric embeddings from an approximation-algorithm perspective. In other words, we would like to address questions of the form: given an input metric, how to best embed it into a prescribed host metric? In particular, we focus on the line metric as the prescribed host m etric. It turns out that many of these problems are NP-hard. Therefore, we look for approximation algorithms.

While distortion as defined above has been very popular, we also investige other notions of the quality of the embedding. We explain each of these below.

Let $(V, d)$ denote the given finite guest metric. We want to embed $(V, d)$ into the line metric: $(\mathbb{R}, \delta)$. Let $|V| = n$ and let $\Delta$ be the diameter of the metric.

## 1.1 Average distortion

In Chapter 2, we focus our attention on the average distortion of the embeddings arbitrary finite metrics into the line metric $\mathbb{R}$. The average distortion is the factor by which the average distance in the metric is stretched.

In a recent work, [Rabinovich, 2003] introduced the notion of average distortion and proved bounds on average distortion of *non-expanding* embeddings into a line. Rabinovich also showed a close connection between this and the max-flow min-cut ratio for concurrent multicommodity flow with applications to finding quotient cuts in graphs ([Leighton and Rao, 1999]).

We prove that finding the best embedding of even a tree metric into a line metric so as to minimize the average distortion is NP-hard, and hence focus on *approximating* the average distortion of the best possible embedding for the given input metric. We give a constant-factor approximation for the problem of embedding general metrics into the line metric. For the case of $n$-point tree metrics, we provide a quasi-polynomial time approximation scheme (QPTAS) which outputs an embedding with distortion at most $(1 + \epsilon)$ times the optimum in time $n^{O(\log n/\epsilon^2)}$. We also consider the average distortion, where the average is taken only over the endpoints of the edges of an input tree metric, we show how to exploit the structure of tree metrics to give an exact solution in polynomial time.

The basic idea is to think of an embedding into the line as a tour on the nodes of the original metric that starts from the leftmost vertex on the line and visit the vertices in order from left to right. Our results build on this simple observation, and demonstrate a close relationship between minimizing average distortion and the related problems of finding short TSP tours [Lawler et al., 1985], minimum latency tours (Blum et al. [1994], Goemans and Kleinberg [1998], Archer and Williamson [2003]), and optimal $k$-repairmen solutions ([Fakcharoenphol et al., 2003a]). In particular, we prove the following results for the average distortion. These results appeared in [Dhamdhere et al., 2004].

1. **Hardness for average distortion:** We prove that the problem of finding the minimum average distortion non-contracting embedding of finite metrics into the line is NP-hard, even when the input metric is a tree metric. The proof proceeds via a reduction from the Minimum Latency Problem on trees [Sitters, 2002].

2. **Constant-factor approximations:** We give an algorithm that embeds any metric $(V, d)$ into the line with average distortion that is within a constant of the minimum possible over all non-contracting embeddings. In fact, we prove a slightly more general bound on non-contracting embeddings into $k$-spiders (i.e., homeomorphs of stars with $k$ leaves). This result uses a lower bound on the minimum average distortion of a non-contracting embedding into a $k$-spider in terms of the minimum $k$-repairmen tour [Fakcharoenphol et al., 2003a] on the metric.

3. **QPTAS on trees:** For tree metrics on $n$ nodes, we give an algorithm for finding a $(1 + \epsilon)$-approximation to the minimum average distortion non-contracting embedding into a line in $n^{O(\log n/\epsilon^2)}$ time. Our algorithm, which appears in Section 2.2, uses

a lower bound on the minimum average distortion related to the TSP tour length and latencies of appropriately chosen segments of an optimal tour. In this way, it extends the ideas of [Arora and Karakostas, 2003] for minimizing latency on trees to the more general time-dependent TSPs [Blum et al., 1994], and provides a QPTAS for the latter problem as well.

4. **Poly-time algorithm for tree-edge distortion** For a tree metric as input, if the minimum average distortion is measured only over the endpoints of the edges of the tree (we call this objective the average tree-edge distortion), then we show that an embedding following a certain Euler tour of the tree is optimal. In Section 2.3, we show how to find this tour in polynomial time by dynamic programming. This result extends some ideas of [Shiloach, 1979].

**Related Work** While our problem appears similar to that of finding the *Minimum Linear Arrangement (MLA)*, for which Rao and Richa [Rao and Richa, 1998] gave an $O(\log n)$ approximation using the notion of spreading metrics, it is subtly different: the MLA problem involves minimizing the average stretch of the edges $\sum_{\{u,v\}\in E} |\pi(u) - \pi(v)|$ under all maps $\pi : V \to [n]$, whereas the mappings in our problem are $f : V \to \mathbb{R}$, and must ensure that $|f(u) - f(v)| \geq d(u, v)\ \forall \{u, v\} \in V \times V$.

The problem of finding *Minimum Latency tours* (a.k.a. the Traveling Repairman problem) is relevant to our discussion in terms of techniques used. In this problem, one is given a metric space $(V, d)$ and a root depot $r \in V$; a repairman starting at $r$ has to visit all $|V| = n$ customers, one at each node of the metric. The goal is to minimize the *average waiting time* of the customers, where the waiting time (or *latency*) of a customer is the sum of the distances of all edges traversed by the repairman before visiting this customer. There are extensions of this problem to the $k$-repairman case, where $k$ repairmen start off at $r$, and the latency of a customer is now the time at which any one of the repairman visits this customer. The version with only one repairman is known to be NP-hard even on a tree [Sitters, 2002], and is MAX-SNP hard in general [Blum et al., 1994]. The first constant-factor approximation for this problem was given by Blum et al. [Blum et al., 1994]; the approximation factor was improved by Goemans and Kleinberg [Goemans and Kleinberg, 1998] to $7.18$, and most recently by Chaudhuri et al. [Chaudhuri et al., 2003] to $3.59$. For the special cases of the latency problem on trees, Arora and Karakostas [Arora and Karakostas, 2003] gave a quasi-polynomial time approximation scheme (QPTAS); similar results were given for the case when the points lie in $\mathbb{R}^d$ for fixed dimension $d$. The $k$-repairmen version of the problem was studied by [Fakcharoenphol et al., 2003a] who show a 16.994-approximation for arbitrary $k$; this was improved to $8.49$ by [Chaudhuri et al., 2003].

Finally, a problem whose objective is the linear combination the cost of a tour as well as its latency is that of finding *time dependent TSP tours*; the paper by Blum et al. [Blum et al., 1994] gives a constant factor approximation algorithm for this problem.

## 1.2 Additive distortion

In chapter 3, we consider he additive distortion of embeddings into the line metric. The additive distortion is the sum of differences in all pairwise distances between the embedded and input distances. The $L_p$ norm of additive distortion is defined as:

$$(\sum_{x,y}|\delta(x,y) - d(x,y)|^p)^{1/p}.$$

The problem of finding the embedding into the line metric with minimum additive distortion was shown to be NP-hard by [Saxe, 1979]. Our main result is $O(\log^{1/2p}(n))$-approximation algorithm for the $L_p$ norm of the additive distortion. This result has appeared in [Dhamdhere, 2004]

It's important to note here that we do not restrict the embedding to be non-contracting. Instead we consider the absolute difference between the distances.

**Related Work** The additive distortion as a measure of the quality of the embedding has received much attention, especially for the numerical taxonomy problem. The numerical taxonomy problem is one of finding a tree metric that closely fits the input metric data. Formulation of this problem as the minimization of additive distortion was first proposed by [Cavalli-Sforza and Edwards, 1967] in 1967. In 1977, Waterman et al. [Waterman et al., 1977] showed that if there is a tree metric $T$ coinciding exactly with the input data $D$, then it can be constructed in linear time. In the case when there is no tree that fits the data perfectly, Agarwala et al. [Agarwala et al., 1999] used the framework of approximation algorithms to give heuristics with provable guarantees for the problem. They gave a 3-approximation to the $L_\infty$ norm of the additive distortion for fitting the data to a tree metric. They reduced the problem to that of fitting the data to *ultrametric*, where each leaf is at the same distance from a common root. For *ultrametrics*, they used an exact polynomial-time algorithm for the $L_\infty$ norm due to Farach et al. [Farach et al., 1995].

In our setting the host metric is the line metric. The special case of the problem for the $L_\infty$ norm (i.e. with $p = \infty$) was considered by Håstad et al. [Håstad et al., 1998]. They gave a 2-approximation for it.

For fitting points to a line, a well-known result due to Menger (see e.g. [Deza and Laurent, 1997]) gives the following four point criterion. The four point criterion says that, if every subset of size $4$ can be mapped into the real line exactly, then all the points can be mapped into the line exactly. An approximate version of Menger's result was given by Badoiu et al. [Bădoiu et al., 2003]. They proved that if every subset of size $4$ can be embedded into the line with the $L_\infty$ norm of the additive distortion being at most $\epsilon$ then all the points can be embedded with the $L_\infty$ norm of the additive distortion being at most $6\epsilon$.

## 1.3 Classical Distortion

In chapter 4, we address the problem of approximating the classical distortion. Given a graph $G = (V, E)$ inducing a shortest path metric $M = M(G) = (V, d)$, find a mapping $f$ of $V$ into a *line* that is non-contracting (i.e., $|f(u) - f(v)| \geq d(u, v)$ for all $u, v \in V$) which minimizes the distortion $D(M, f) = \max_{u,v\in V} \frac{|f(u)-f(v)|}{d(u,v)}$. That is, our goal is to find $D(M) = \min_f D(M, f)$. For the case when $G$ is an *unweighted* graph, we show the following algorithms for this problem (denote $n = |V|$):

- A polynomial $O(D)$-approximation algorithm for metrics $M$ for which the optimal distortion is $D$. This also implies an $O(\sqrt{n})$-approximation algorithm for any $M$.
- A polynomial-time $\tilde{O}(\sqrt{D})$ approximation algorithm for metrics generated by unweighted trees. This also implies an $\tilde{O}(n^{1/3})$-approximation algorithm for these metrics.

Most of these results have appeared as part of [Bădoiu et al., 2005b].

For a special case in unweighted trees we give an improved $O(\log n)$-approximation.

**Related Work** Recently, Kenyon, Rabani and Sinclair [Kenyon et al., 2004] gave *exact* algorithms for minimum (multiplicative) distortion embeddings of metrics *onto* simpler metrics (e.g., line metrics). Their algorithms work as long as the minimum distortion is small, e.g., constant. We note that constraining the embeddings to be *onto* (not *into*, as in our case) is crucial for the correctness of their algorithms.

Very recently Badoiu et al. [Bădoiu et al., 2005a] gave an $o(n)$-approximation algorithm for embedding weighted graphs into the line metric. They also showed that it is NP-hard to approximate it within $n^\delta$ for some small constant $\delta > 0$.

## 1.4 Weighted Bandwidth

In chapter 5, we consider the problem of finding a linear ordering that minimizes the stretch. In other words, given a metric $(V, d)$, we want to map the points to $\{1, 2, \ldots, n\}$, so as to minimize $max_{x,y}|f(x) - f(y)|/d(x, y)$. i.e., instead of non-contracting embedding, we look for just a linear ordering. We give an $O(\log^2(n) \log \Delta)$-approximation for this problem, where $\Delta$ is the diameter of the metric. As a generalization of this result, we also get an approximation algorithm for the weighted bandwidh problem. Weighted bandwidth is defined $max_{x,y}|f(x) - f(y)|w(x, y)$, where $w(x, y)$ denotes the weight of the edge $(x, y)$. Our approximation guarantee for the weighted bandwidth problem is $O(\log^2 n \log nW)$, where $W$ is the maximum weight of an edge. These results appear in [Dhamdhere, 2005].

**Related Work** The (unweighted) bandwidth minimization problem (i.e. when all the edge weights are $1$) arises in VLSI layout problems and has received much attention. It was shown to be NP-hard by Papadimitriou [Papadimitriou, 1976]. Blum et al. [Blum et al., 2000] gave an SDP relaxation of the bandwidth and obtained an $O(\sqrt{n/b^*})$ approximation, where $b^*$ is the optimal bandwidth. The first non-trivial approximation to this problem was given by Feige [Feige, 2000]. He developed a notion of *volume-respecting* embedding and used it to give $O(\log^{4.5} n)$-approximation for the bandwidth problem. Subsequently, Dunagan and Vempala [Dunagan and Vempala, 2001] showed how to improve the approximation factor based on the SDP relaxation of Blum et al. [Blum et al., 2000]. Recently, Krauthgamer et al. [Krauthgamer et al., 2004] showed an algorithm to construct volume respecting embeddings and thus reduced the approximation factor to $O(\log^3 n)$.

## 1.5 Embeddings into spanning trees

In chapter 6, we study probabilistic embeddings of graphs into induced spanning trees. Given a graph $G = (V, E)$, we consider the shortest path metric on it defined by $(V, d)$. A probabilistic embedding into induced spanning trees is a probability distribution over the spanning trees of graph $G$. The quality of the embedding is measured by *expected distortion*, which is the maximum over the edges of $G$, of the expected value of the distance between its endpoints in the spanning tree.

Only in this chapter, we are interested in uniform bounds on the expected distortion.

**Related Work** The problem of embedding a graph into a spanning tree to minimize the average distortion was first considered by Alon, Karp, Peleg and, West [Alon et al., 1995]. They gave an algorithm to construct a spanning tree with $O(exp(\sqrt{\log n \log\log n}))$ average distortion and applied to the online $K$-server problem. They also demonstrated examples where $\Omega(\log n)$ average distortion would be incurred for any spanning tree.

Subsequently, Bartal [Bartal, 1996, 1998] considered the problem of probabilistic embeddings of arbitrary metrics into tree metrics (not necessarily spanning trees). He obtained $O(\log^2 n)$ expected distortion and subsequently improved it to $O(\log n \log\log n)$. He also proved a lower bound of $\Omega(\log n)$ for expander graphs. Later Fakcharoenphol, Rao and Talwar [Fakcharoenphol et al., 2003b] gave an algorithm with $O(\log n)$ expected distortion, thus matching the lower bound.

The case of embedding into *induced* spanning trees was still open. Recently, Emek and Peleg [Emek and Peleg, 2004] gave an $O(\log n)$-approximation algorithm for minimizing the distortion of a single spanning tree.

In 2004, Spielman and Teng [Spielman and Teng, 2004] showed that embedding into a spanning tree with average stretch $\phi$ yields an $O(m\phi \log^{O(1)} n)$-time algorithm for solving diagonally-dominant symmetric linear systems. Subsequently, Elkin, Emek, Spielman and Teng [Elkin et al., 2005] made a breakthrough for the average distortion problem. Their algorithm had $O(\log^2 n \log\log n)$ average distortion.

**Our results** We give a simple algorithm with $O(\log^2 n)$ expected distortion. This also implies an $O(\log^2 n)$ bound on average distortion. Our algorithm using the star-decomposition schema introduced by Elkin et al. [Elkin et al., 2005]. We combine it with the cutting scheme of Bartal [Bartal, 1996]. Furthermore, we introduce a new technique, viz. tail-bounds on the diameter of the resulting sub-trees to bound the distortion. Our techniques are orthogonal to those used by Elkin et al.. It might be possible to improve upon our results by combining these ideas. These results also appear as part of [Dhamdhere et al., 2006].

# Chapter 2

# Average Distortion

It is important to note that while any non-contracting embedding can be converted to a non-expanding embedding with the same average distortion by scaling down all the distances, the converse is not true. Indeed, a non-expanding embedding $f$ might not be one-one, and may map two points in the guest metric to the same point in the host metric. This is a crucial difference between the two problems, and hence our result does not give a constant-factor approximation for the average distortion of non-expanding embeddings into the line $\mathbb{R}$.

## 2.1 Embedding arbitrary metrics into the line

In this section, we show that we can approximate the average distortion into a line for a given metric to within a constant; to this end, we show that the problem is closely related to that of finding the minimum latency tours and its generalizations in a finite metric space.

### 2.1.1 Hardness of Embeddings

**Theorem 1** *It is NP-hard to find a non-contracting embedding of a given metric induced by a tree into a line that minimizes the average distortion.*

**Proof.**

We show how to reduce the problem of finding minimum latency tour on trees to our problem. The minimum latency problem on trees (tree-MLP) was shown to be NP-hard by Sitters [Sitters, 2002] even when the edge lengths are in $\{0, 1\}$.

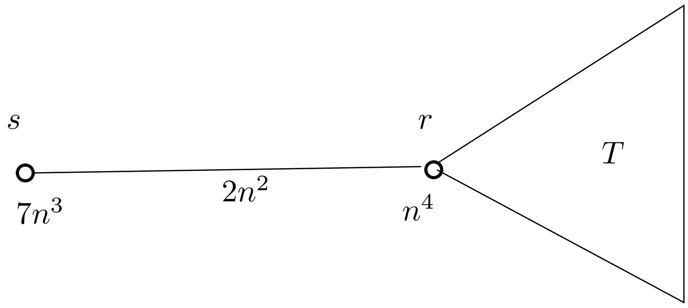


Figure 2.1: Hardness construction

Given an instance of tree-MLP, our reduction will define an instance of the average distortion problem on a tree where the vertices have integer weights and the edges have lengths, and we generalize the definition of average distortion to be

$$\rho_w(f) = \frac{\sum_{x,y \in V} w_x w_y \delta(x,y)}{\sum_{x,y \in V} w_x w_y d(x,y)}. \tag{2.1}$$

As long as the weights are only polynomially bounded, we can convert such an instance to one with unit vertex-weights by the simple expedient of replacing any vertex with weight $w$ by a set of $w$ vertices at distance zero from one another. Let us also note that minimizing the average distortion is equivalent to minimizing total distance in the embedding, and hence we will show the hardness of minimizing the total distance.

Given a tree $T$ rooted at $r$ as an instance of tree-MLP problem with edge lengths in $\{0, 1\}$, we construct an instance of the average distortion problem (cf. Figure 2.1). We introduce a new vertex $s$ and connect it to the root $r$. We assign weight $7n^3$ to $s$ and $n^4$ to $r$. Let the distance between $r$ and $s$ be $d_{r,s} = 2n^2$. The rest of the vertices have weight $1$.

**Claim 2.1** *In the optimal embedding, $r$ and $s$ are adjacent to each other.*

**Proof.** Consider any embedding in which $r$ and $s$ are not adjacent to each other. Therefore, the distance between $r$ and $s$ is at least $d(r,s) + 1$ in such an embedding. The total distance in this embedding is at least $w_r w_s \cdot (d(r,s) + 1) = 14n^9 + 7n^7$.

On the other hand, consider any embedding in which $r$ and $s$ are adjacent to each other and the pairwise distance between adjacent pairs is same as that in the guest tree metric. We now compute an upper bound on the total distance in such an embedding. The contribution due to the pair $(r, s)$ is $w_r w_s \cdot d(r, s) = 14n^9$. The contribution due to the pairs of the form $(r, v_i)$ or $(s, v_i)$ is at most $(w_r + w_s) \cdot (2d(r, s) + n^2) \cdot n \leq 6n^7$, since the distance between any two points in the embedding is at most $2d(r, s) + n^2$. Finally, the contribution from the pairs $(v_i, v_j)$ is at most $n^2 \cdot (2d(r, s) + n^2) \leq 5n^4$. Thus the total contribution is at most $12n^8 + 6n^7 + 5n^4$.

Therefore, any embedding in which $r$ and $s$ are adjacent is better than any embedding in which they are not. Therefore, in any optimal embedding, $r$ and $s$ have to be adjacent to each other. ∎

**Claim 2.2** *In any optimal embedding, no vertex $v_i$ and the vertex $s$ are on the same side of $r$.*

**Proof.** Suppose that the vertex $v_i$ and $s$ are on the same side of $r$. From the previous claim it follows that $s$ must be between $v_i$ and $r$. Therefore the pair $(v_i, r)$ contribute at least $w_r d(r, s)$ to the total distance. Now we construct an alternative embedding the current one. We keep the order of all the vertices except $v_i$ the same. We embed $v_i$ on the opposite side of $r$ at the end. In this process only the contributions from the pairs $(v_i, v_j)$ for all $j$ and $(v_j, s)$ go up, while the contribution from the pair $(v_i, r)$ goes down. Note that, in the new embedding, the contribution of the pairs $(v_i, v_j)$ can be at most $(2d(r, s) + n^2) \cdot n$ and the contribution of the pair $(v_i, s)$ is at most $w_s(2d(r, s) + n^2)$. The contribution due to the pair $(v_i, v_r)$ goes down by at least $w_r \cdot d(r, s) - w_r \cdot n^2$. Adding up the changes in contributions, we get that the new embedding has smaller total distance.

Therefore in any optimal embedding, the vertices $v_i$ and $s$ cannot be on the same side of $r$. ∎

In order to finish the proof of the theorem, we now show that the ordering of the vertices in an optimal tree-MLP tour is $r, v_1, v_2, \ldots, v_n$ if and only if $s, r, v_1, v_2, \ldots, v_n$ is the ordering of the vertices in the embedding that minimizes the average distortion. Let $s, r, \pi(1), \pi(2), \ldots, \pi(n)$ be the ordering of the vertices in an embedding. Let $L(\pi)$ denote the total latency of the ordering given by $r, \pi(1), \ldots, \pi(n)$. Let $Av(\pi)$ denote the sum of the distances in the embedding consisting of $\pi(1), \ldots, \pi(n)$ in that order.

Then, the total distance in the embedding is

$$w_s \cdot w_r + w_s \cdot n + (w_s + w_r) \cdot L(\pi) + Av(\pi)$$

Note that $Av(\pi)$ is bounded above by $n^4$ since we sum the distances over $\binom{n}{2}$ pairs and the maximum distance between any pair $\{v_i, v_j\}$ in the embedding is at most $n^2$. Thus, $Av(\pi)$ is smaller than $(w_r + w_s)$.

Note that the difference between optimal value of $L(\pi)$ and that in any other solution is at least 1, while it's multiplying factor $(w_r + w_s)$ dominates $Av(\pi)$. Hence, in order to minimize the total distance, we have to minimize $L(\pi)$. This is exactly the tree-MLP problem. Hence, the problem of minimizing the average distortion is NP-hard. ■

### 2.1.2 A Constant-factor Approximation Algorithm

In order to make the exposition of our approximation algorithm simple, we first show a simple 2-approximation for embedding a given metric into trees. Then we consider embeddings into $k$-spiders and show how a similar technique works for them (a $k$-spider is a tree with all vertices except the *center* having degrees 1 or 2, and hence is a homeomorph of the star with $k$ leaves). In particular, we show how to take a $\rho$-approximation algorithm for the $k$-repairmen problem [Fakcharoenphol et al., 2003a], and use it to produce a $2\rho$-approximation for average distortion of embedding a given metric into a $k$-spider. Finally, since a line metric is equivalent to a 2-spider, we get the embedding into a line metric as a corollary.

**Embeddings into trees** Consider the problem of embedding the given metric $d$ into a tree metric $\delta$ to minimize average distortion. Let $\Delta = \sum_{x,y \in V} d(x,y)$ denote the sum of all the distances in the metric $d$, and hence $\mathrm{av}(d) = \Delta/n^2$ is the average distance in $d$. The *median* of the metric $d$ is the point $v \in V$ that minimizes $\Delta_v = \sum_{w \in V} d(v,w)$, and will be denoted by *med*. Note that we can decompose $\Delta$ as follows:

$$\Delta = \sum_{u,v \in V} d(u,v) = \sum_{u \in V} (\sum_{v \in V} d(u,v)) = \sum_{u \in V} \Delta_u \geq n\, \Delta_{med} \tag{2.2}$$

since $\Delta_{med} \leq \Delta_v$ for all $v \in V$. Consider a shortest-path tree $T$ (which is a star in a general metric $d$) rooted at *med*, and let $d_T$ denote the metric induced by this shortest path tree. Then the total distance in this tree $T$ is

$$\begin{aligned}\Delta_T = n^2 \cdot \text{av}(d_T) &= \sum_{u,v \in V} d_T(u,v) \leq \sum_{u,v \in V} d_T(\text{med}, u) + d_T(\text{med}, v) \\ &= \sum_{u,v \in V} d(\text{med}, u) + d(\text{med}, v) = 2n\Delta_{med}\end{aligned}$$

where the inequality in the second step is just the triangle inequality. This implies that $n\Delta_{med} \leq \Delta \leq \Delta_T \leq 2n\Delta_{med}$, and thus:

**Lemma 1 ((See also [Wong, 1980]))** *Given any graph, the total distance $\Delta_T$ for the shortest path tree rooted at the median is at most $2\,\Delta$, and is a $2$-approximation for the problem of embedding the graph into trees.*

The bound of $2$ is tight. E.g. in a complete graph the total distance is $n(n-1)$ and it is $n(2n-3)$ for the shortest path tree. Also note here that the bound of 2 above is an *absolute* bound on the worst-case ratio between the average distance in the output tree and the graph, and is in the same flavor as the more traditional results on bounding the maximum distortion of embeddings. We next move toward an approximation approach by restricting the class of trees into which we embed.

**Embeddings into spiders** We now generalize the previous result to the case of embeddings into $k$-spiders. The vertex of degree $k$ is called the *center* of the spider, and the components obtained by removing the center are called its *legs* [Klein and Ravi, 1995].

Let $d_k^*$ denote the optimal $k$-spider embedding. We decompose the sum of distances in $d_k^*$ as the sum of $k$-repairman path rooted at each vertex. Recall that, in $k$-traveling repairman problem, we are given $k$ repairmen starting at a common depot $s$. The $k$ repairmen are to visit $n$ customers sitting one per node of the input metric space. The goal is to find tours on which to send the repairmen so as to minimize the total time customers have to wait for a repairman to arrive [Fakcharoenphol et al., 2003a].

Let $c$ be the center of the spider in the optimal $k$-spider embedding. To construct a $k$-repairman paths starting from a vertex $r$, we do the following. We send one repairman away from the center along the leg of the spider which contains $r$. The other $k-1$ repairmen travel toward the center $c$ of the spider. From the center, they go off, one per remaining leg of the spider. The cost of this $k$-repairman tour is $\Delta_r^* = \sum_j d_k^*(r,j)$. Summing over all choices of the root we see that this is same as the sum of distances in the embedding $d_k^*$.

$$n^2 \cdot \text{av}(d_k^*) = \sum_{u,v \in V} d_k^*(u,v) = \sum_{v \in V} \Delta_v^*$$

Hence, $n$ times the cost of the cheapest $k$-repairman tour over all choices of the depots (denoted by $\Delta^{opt}$), is a lower bound on the sum of all the distances. i.e.,

$$\sum_{u,v \in V} d_k^*(u,v) \geq n \cdot \min_r \{\Delta_r^{opt}\}.$$

Consider the cheapest $k$-repairman tour over all choices of centers. Let it be centered at a vertex $c$. This tour defines a non-contracting embedding into a $k$-spider with $c$ at the center of the spider. Let $d^c(u)$ denote the distance of vertex $u$ from the center $c$ in the tour. We can bound the sum of distances in this embedding as follows:

$$\sum_{u,v \in V} d_k^c(u,v) \leq \sum_{u,v \in V} d^c(u) + d^c(v) \leq 2n \sum_{u \in V} d^c(u) \leq 2 \sum_{u,v \in V} d_k^*(u,v).$$

Thus, if we could compute the optimal $k$-repairman tour centered at $c$ exactly, we would obtain a 2-approximation to the problem of embedding the metric into $k$-spiders. Although the problem of finding an optimal $k$-repairman tour is NP-hard, the argument above proves the following.

**Theorem 2** *Given a $\gamma$-approximation for the minimum $k$-repairmen problem on a metric $d$, we can obtain a $2\gamma$-approximation for embedding the metric $d$ into a $k$-spider in a non-contracting fashion to minimize the average distortion.*

The current best known approximation factor for the $k$-repairman problem is $8.49$ (due to Chaudhuri et al. [Chaudhuri et al., 2003]), leading to the following corollary.

**Corollary 2.3** *There is a 16.98-approximation for minimizing the average distortion of a non-contracting embedding of a given finite metric into a $k$-spider.*

## 2.2 Approximation Schemes for trees

In this section, we restrict our attention to the special case of tree metrics. We give a quasi-polynomial time approximation scheme (QPTAS) for minimizing the average distortion for embeddings into the line metric. Our algorithm is based on the QPTAS given by Arora and Karakostas [Arora and Karakostas, 2003] for the minimum latency problem. They proved that a near-optimal latency tour can be constructed by concatenating $O(\log |V|/\epsilon)$ optimal Traveling Salesman paths, and the best such solution can be found by dynamic programming.

For an embedding $f : V \to \mathbb{R}$ into the line, let the *span* of the embedding be defined as $\max_{x,y} |f(x) - f(y)|$, the maximum distance between two points on the line. We note that an embedding with the shortest span is just the optimal Traveling Salesman path. While embedding a given metric into the line metric, minimizing the span of the embedding could result in very high average distortion. However, we show that it suffices to minimize the span locally to find near optimal embedding. In particular, our solution within $(1 + \epsilon)$ of optimal minimum average distortion is to find an embedding that is the union of $O(\log |V|/\epsilon^2)$ Traveling Salesman paths with geometrically decreasing number of vertices.

In the sequel, we use $n$ to denote $|V|$, the number of vertices. For our algorithm, we assume that all the edge lengths are in the range $[1, n^2/\epsilon]$. Indeed, if $D$ is the diameter of the metric space and $u$ and $v$ are two vertices such that $d(u, v) = D$, then $\sum_{x,y\in V} d(x, y) \geq \sum_{x\in V} d(x, u) + d(x, v) \geq nD$. We can then merge all pairs of nodes with inter-node distance at most $\epsilon D/n^2$, which affects the sum of distance by at most $\epsilon nD$. Hence the ratio of maximum to minimum nonzero distance in the metric can be assumed to be $n^2/\epsilon$.

**Relation to TDTSPs** We first show that the Arora-Karakostas QPTAS works also for the case of the Time Dependent Traveling Salesman Problem (TDTSP) defined by Blum et al. [Blum et al., 1994]. In the TDTSP, the objective is to minimize a positive linear combination of the TSP tour value and the total latency of the tour. The objective function is of the form $\alpha\mathsf{TSP} + \beta\mathsf{LAT}$ where $\mathsf{TSP}$ and $\mathsf{LAT}$ denote the span of the tour and total latency of the tour respectively and $\alpha$ and $\beta$ are constants.

We now describe how to break up an optimal tour into locally optimal segments. Let $\mathcal{T}$ denote the optimal tour for the objective function $\alpha\mathsf{TSP} + \beta\mathsf{LAT}$. We break this tour into $k$ segments ($k$ is $O(\log n/\epsilon)$). In segment $i$ we visit $n_i$ nodes, where

$$n_i = \lceil (1 + \epsilon)^{k-1-i} \rceil \text{ for } i = 1, \ldots, k - 1; \qquad n_k = \lceil 1/\epsilon \rceil$$

Note that these $n_i$'s are chosen in such a way that $n_i \leq \epsilon \sum_{j>i} n_j$. Denote $\sum_{j>i} n_j$ by $r_i$. Replace the optimal tour in each segment, except the last one, by the minimum-distance traveling salesman path on the vertices of that segment that starts and ends at the same pair of vertices.. The new tour now consists of the concatenation of $O(\log n/\epsilon)$ locally optimal Traveling Salesman paths. This gives us the following lemma.

**Lemma 2** *There is a tour that is a concatenation of $O(\log n/\epsilon)$ minimum Tr aveling Salesman paths that has $\alpha\mathsf{TSP}+\beta\mathsf{LAT}$ objective value at most $(1+\epsilon)$ times the optimal solution (OPT).*

**Proof.** We first give a lower bound on OPT. Let $T_i$ denote the span of the segment $i$ in OPT. Every node in the $m^{\text{th}}$ segment has latency bigger than $\sum_{j=1}^{m-1} T_j$. We sum over all vertices and get the lower bound on OPT: OPT $\geq \sum_{i=1}^{k-1}(\alpha + \beta r_i)T_i$.

Now we replace each segment of OPT with the minimum Traveling Salesman path on the same set of vertices with the same pair of vertices as start and end points. By replacing a segment with a minimum traveling salesman path, we reduce the span of that segment. However latency of the vertices inside a segment can go up. The latency of each vertex in $i^{\text{th}}$ segment will increase by at most $n_i T_i$. Hence the cost of concatenated tour increases by at most $\sum_{i=1}^{k-1} \beta n_i T_i$. From the property that $n_i \leq \epsilon \cdot r_i$, it immediately follows that the cost of the concatenated tour is at most $(1 + \epsilon)$OPT. ■

We now use the Lemma 2 to show the following theorem for average distance.

**Theorem 3** *Any finite metric has a non-contracting embedding into a line that is composed of $O(\log n/\epsilon^2)$ minimum Traveling Salesman path segments with average distortion no more than $(1 + \epsilon)$ times the minimum possible over all such embeddings.*

**Proof.** Our strategy is same as in Lemma 2. Consider the optimal embedding of the input tree into a line. We break this embedding up into $O(\log n/\epsilon)$ segments. Let $n_i$ be the size of $i^{\text{th}}$ segment defined as before. We now divide the objective function value according to the segments, so that only the share $C_i$ of segment $i$ changes, if we replace the embedding of segment $i$ with a different embedding.

Let $T_i$ be the span of the embedding of segment $i$. If $i_0$ is the left-most node in the embedding of the segment $i$, then let $L_i = \sum_{j \in n_i} \delta(i_0, j)$ be the sum of the distances of all nodes in segment $i$ from node $i_0$. Note that $L_i$ is the total latency of vertices in segment $i$ with $i_0$ as root. And let $D_i = \sum_{u,v \in n_i} \delta(u, v)$ be the sum of all the pairwise distances in segment $i$.

Let $q_i = \sum_{j<i} n_j$ and $r_i = \sum_{j>i} n_j$ be the number of total nodes to the left and right of segment $i$ respectively.

We now describe a lower bound on the total distance of the optimal solution. We define the contribution of the segment $i$ to the lower bound as the sum of the following distinct terms.

1. If a vertex $u$ is to the left of the segment $i$ and a vertex $v$ is to the right, then the segment $i$ adds $T_i$ to the distance between them.

2.If a vertex $u$ is to the left and $w$ is in the segment $i$, then the contribution is $\delta(i_0, w) =$ the distance from the left most vertex $i_0$ of the segment $i$ to $w$.

3.If a vertex $v$ is to the right and $w$ is in the segment $i$, then the contribution is $T_i - \delta(i_0, w)$.

4.If both the vertices $w$ and $w'$ are in the segment $i$, then the contribution is $\delta(w, w')$.

These contributions, when summed up over all pairs of vertices, give:

$$C_i = q_i r_i T_i + q_i L_i + r_i(n_i T_i - L_i) + D_i \tag{2.3}$$

Note that $\sum_i C_i$ is a lower bound on the total distance. In the following argument we rearrange the embedding inside each component while making sure that the increase in the total distance is at most $\epsilon \sum_i C_i$.

Note that $D_i \leq n_i^2 T_i$. For $i = 2, \ldots, k$, we know that $n_i \leq q_i$ and $n_i \leq \epsilon \cdot r_i$. Hence, comparing $D_i$ with the first term in (2.3), we get

$$(1+\epsilon)(q_i r_i T_i + q_i L_i + r_i(n_i T_i - L_i)) \geq C_i \geq q_i r_i T_i + q_i L_i + r_i(n_i T_i - L_i) \tag{2.4}$$

To prove the Theorem 3, it suffices to find an embedding of the $i^{\text{th}}$ segment such that the increase in the total distance is within $\epsilon$ times the lower bound in the RHS of the above inequality 2.4. The expression for the lower bound on the RHS of inequality 2.4 is a linear combination of TSP and Latency values of the tour in segment $i$. We can apply Lemma 2 to obtain a tour composed of $O(\log n_i/\epsilon)$ minimum traveling salesman paths. Note that replacing the original embedding with the tour obtained from Lemma 2 can only increase the four distinct terms that make up the quantity $C_i$. From Lemma 2, the increase in the total distance is at most $\epsilon C_i$.

A technical detail in this argument is that the coefficient of $L_i$ could be negative. Lemma 2 does not handle this case. But note that $n_i T_i - L_i$ is the total "reverse" latency in segment $i$ with the rightmost endpoint being the root. Thus we can rewrite the lower bound as a linear combination of $T_i$ and $n_i T_i - L_i$ with positive coefficients.

We can thus replace each segment $i$, with a concatenation of $O(\log n_i/\epsilon)$ Traveling Salesman paths, without increasing the cost by more than a factor of $(1+\epsilon)$. Since there are $O(\log n/\epsilon)$ segments in all, it follows that there is an embedding consisting of $O(\log^2 n/\epsilon^2)$ shortest Traveling Salesman paths.

Finally, we show how to reduce this number down to $O(\log n/\epsilon^2)$. Let us rewrite the lower bound in (2.4) as $(q_i - r_i)L_i + (q_i + n_i)r_i T_i$. Note that $L_i \leq n_i T_i$.

This gives us that the term $(q_i - r_i)L_i$ is at most $\epsilon \cdot (q_i + n_i)r_iT_i$, whenever $q_i - r_i$ is positive. Hence, if we replace the segment $i$ with a shortest Traveling salesman path on those vertices, the cost will be within $(1 + \epsilon)$ of the lower bound in (2.4). Note that, for $i \geq 1/\epsilon$, we have $q_i \geq r_i$. Hence for $i = 1, \ldots, 1/\epsilon$, using Lemma 2, we replace each segment by a concatenation of $O(\log n/\epsilon)$ tours each. Then for the segments $1/\epsilon$ and above, we use only one minimum Traveling Salesman path per segment. Overall this results in a concatenation of $O(\log n/\epsilon^2)$ traveling salesman paths with the average distortion within $(1 + \epsilon)$ times that of the optimal. ■

Consider a $(\frac{1}{3}, \frac{2}{3})$-partition of the tree, i.e. a recursive partition of the tree into two subtrees with a common root, such that for each subtree

$$\frac{1}{3} \cdot n \leq (\text{size of subtree}) \leq \frac{2}{3} \cdot n.$$

It is a folklore result that a $(\frac{1}{3}, \frac{2}{3})$-partition exists for any tree. We will use the term *separator node* for the common root of the subtrees. From the recursive partition, we get separator nodes for each level of recursion.

Note that an optimal traveling salesman path on a tree is obtained by depth-first search. Therefore, it need to cross any separator node at most twice. In the previous theorem, we proved that a near-optimal non-contracting embedding is given by a concatenation of $O(\log n/\epsilon^2)$ traveling salesman paths. Combining this with the recursive partition, we get the following theorem.

**Theorem 4** *There exists a non-contracting embedding of a tree metric into a line with average distortion at most $(1 + \epsilon)$ times the minimum possible that, when viewed as a walk, crosses each separator node $O(\log n/\epsilon^2)$ times in a recursive node-separator based partition defined above.*

Using this theorem, we give a dynamic programming algorithm. This is very similar to the algorithm due to Arora and Karakostas [Arora and Karakostas, 2003].

**Theorem 5** *For any given $\epsilon > 0$, there is an algorithm that runs in time $n^{O(\log n/\epsilon^2)}$ and computes a non-contracting embedding of a given input tree metric into a line with average distortion at most $(1 + \epsilon)$-times the minimum.*

**Proof.** Let us describe the dynamic program at the heart of our quasi-polynomial time approximation scheme.

ALGORITHM

"Guess" the leftmost vertex in the embedding. Find a recursive $(\frac{1}{3}, \frac{2}{3})$-partition of the tree. Do the following steps starting at the bottom level of the partition and working upwards.

1. Identify a separator node at the current level of the partition.
2. "Guess" the number of times the embedding crosses this node and for each crossing, the length of the embedding after the crossing and the number of nodes on that portion.
3. Search the dynamic programming table for subtours consistent with the "guesses".
4. Combine the subtours found to create a new bigger subtour and store it in the dynamic programming table and go to step 1.

"Guessing" in step 2 refers to exhaustive enumeration of all possible values for the triple (# of crossings, length, # of nodes). At the end of the enumeration, the algorithm will have created a collection of candidate solutions, one for each possible guess. Its output will be the embedding of minimum average distortion. One of the embeddings considered by this algorithm must be near-optimal. Hence the embedding produced by the algorithm is a $(1+\epsilon)$ approximation for the optimal average distortion.

We now prove that the running time of the algorithm is bounded by $n^{O(\log n/\epsilon^2)}$. The running time is dominated by the number of "guesses". The number of crossings through a node is at most $O(\log n/\epsilon^2)$ and the number of nodes visited between two crossings cannot be greater than $n$. To bound the number of guesses for the length of the embedding between two crossings, we round the lengths as follows. Let $L$ be the length of the longest path in the input tree. We merge all the pairs of vertices with pairwise distance smaller than $\epsilon L/n^3$. We also round each edge length to its closest multiple of $\epsilon L/n^3$ and divide all the lengths by $\epsilon L/n^3$. In this rounded instance, the minimum length is $1$, while the maximum internode distance is $n^3/\epsilon$. After solving the rounded instance, we reinstate the merged edges to the output embedding. This does not change the pairwise distance between any pair by more than $O(\epsilon L/n^2)$. Thus the total change due to rounding is bounded by $O(\epsilon L) = O(\epsilon OPT)$.

If we run the algorithm on a rounded instance, the total number of guesses for each crossing is $O(n^3/\epsilon)\cdot n = O(n^4/\epsilon)$. This gives a total of $O(\frac{\log n}{\epsilon^2}\cdot(n^4/\epsilon)^{O(\log n/\epsilon^2)} = n^{O(\log n/\epsilon^2)}$ guesses for a node. We do this for each node. Moreover, there are $n$ choices for the leftmost vertex of the embedding. Therefore, the overall running time of the algorithm is bounded by $O(n \cdot n \cdot n^{O(\log n/\epsilon^2)}) = n^{O(\log n/\epsilon^2)}$. ■

## 2.3 An exact algorithm for minimizing average tree-edge distortion

For the tree metrics, we consider a slightly different objective function in this section. Let $M = (V, d)$ be the input metric to be embedded in a non-contracting mapping to a line. Assume that the input metric $M$ arises from a tree $T = (V, E)$. Instead of considering distances between all pairs of nodes, we take average of the distance over the edge set $E$ of the tree. Let $l$ denote the host metric (i.e. a line). Then we want to minimize $\sum_{(u,v)\in E} \delta(u, v)$. We call this the *average tree-edge distortion*. We give a polynomial time algorithm for minimizing the average tree-edge distortion.

This problem is quite similar to the Minimum Linear Arrangement [Shiloach, 1979] problem on trees. Recall that, a linear arrangement of a graph $(V, E)$ is a mapping $\pi : V \to [n]$. The objective is to minimize $\sum_{\{u,v\}\in E} |\pi(u) - \pi(v)|$. However, the crucial difference is that we require the embedding into a line to be non-contracting.

Our algorithm is based on the algorithm for Minimum Linear Arrangement on trees given by Shiloach [Shiloach, 1979] with some crucial extensions. We first begin by finding a centroid of the tree. The following lemma is folklore (see, e.g., [Buckley and Harary, 1990]). It is important to note that, we allow subdivision of the edges here, i.e. we allow to split an edge into two by adding a vertex anywhere along that edge.

**Lemma 3** *Given a tree $T = (V, E)$ with edge weights, there exists a centroid vertex $v^*$ in a subdivision of $T$, such that the subtrees of $T$ rooted at $v^*$ have edge weight at most half the total weight of the tree.*

We then show that the subtrees of the centroid are not interleaved in an optimal embedding. This lets us solve the problem recursively on the subtrees. The algorithm constructs an Eulerian tour of the tree as an optimal embedding.

### 2.3.1 Cost reducing transformations

We now show that, given a non-contracting embedding of a tree into the line, we can transform it *without increasing the average distortion*, so that the solutions for subtrees rooted at the centroid are disjoint contiguous segments of the line. We will denote the embedding by a permutation $\pi$ of the vertices. Note that for the embedding to be non-contracting, it suffices to have the distance between adjacent pair of vertices in the permutation to be the same as their distance in the tree metric (i.e. $\delta(i, i + 1) = d(\pi^{-1}(i), \pi^{-1}(i + 1))$.

We now explain the transformations. Let $T$ be the input tree with $v^*$ as the centroid. Let $T_1$ be a subtree of $T$ rooted at $v^*$. We group all other subtrees as $T_2$ (see Figure 2.2). The transformations work toward uninterleaving the embeddings of $T_1$ and $T_2$. There are two different cases depending on whether end vertices are from same or different subtrees.

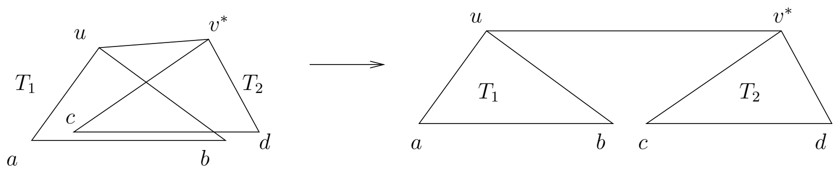


Figure 2.2: Type **A** transformation

1. Let the two endpoints be in different subtrees, i.e. we have $\pi^{-1}(1) \in T_1$ and $\pi^{-1}(n) \in T_2$. A transformation of type $A$ converts the ordering $\pi$ into $\pi_a$, such that $\pi$ restricted to each of $T_1$ and $T_2$ is preserved, and $T_1$ is embedded entirely to the left of $T_2$; i.e., $\pi_a(u_i) < \pi_a(v_j)$ for all $u_i \in T_1$ and $v_j \in T_2$.

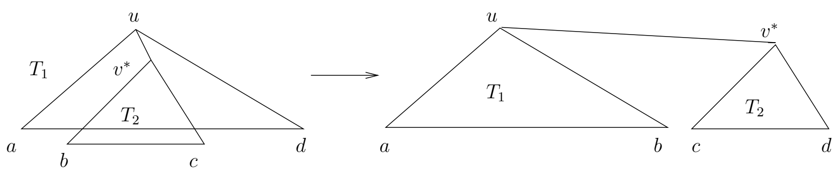


Figure 2.3: Type **B** transformation

2. Let the two endpoints of the embedding be in the same tree, i.e. $\pi^{-1}(1), \pi^{-1}(n) \in T_1$. A type $B$ transformation produces an ordering $\pi_b$ which is same as $\pi$ when restricted to each of $T_1$ and $T_2$. We have two choices: $T_1$ or $T_2$ could be embedded to the left of the other subtree. We pick the one minimizing average tree-edge distortion.

We denote the embedding produced by $\pi_a$ or $\pi_b$ by $(T_1 : T_2)$. Note that, there are two choices for the embedding of $T_1$ (resp. $T_2$): the same order as in $\pi$ or completely opposite of $\pi$. We will always pick the best of these choices.

**Observation 2.4** *In the embeddings $\pi_a$ and $\pi_b$, the length of the edges within the trees $T_1$ and $T_2$ is never more than their counterparts in the embedding $\pi$.*

**Lemma 4** *The above two transformations do not increase the average tree-edge distortion of the embedding.*

**Proof.** We will handle the two cases separately. We need the property that $v^*$ is a centroid vertex only in the second case.

**Type A** The only edge that possibly gets expanded in this transformation is $(v^*, u)$. We show that the increase for this edge is offset by the savings in the edges of the trees $T_1$ and $T_2$. In particular, if $\pi(u_i) > \pi(v^*)$ for $i = 1, \ldots, k$, then in $\pi_a$, the vertices $u_1, \ldots, u_k$ contribute to the cost of edge $(v^*, u)$. However, in the initial ordering $\pi$, these vertices contribute at least this amount to the edges on the path $v^* \to \pi^{-1}(n)$. A symmetric argument holds for the change in the sum of edge lengths in the tree $T_1$.

**Type B** Let $|T|$ denote the length of an Euler tour of tree $T$. We first compute the length of the edge $(u, v^*)$ in $\pi_b$. Since we have picked the cheaper of the two available choices, the length is at most $(|T_1| + |T_2|)/2 + d(u, v^*)$. Thus the potential increase in the length of the edge $(u, v^*)$ is $(|T_1| + |T_2|)/2$. The decrease in the sum of edge lengths of the subtree $T_1$ due to the transformation is at least $|T_2| + d(u, v^*)$. To see this, consider a path $\pi^{-1}(1) \to \pi^{-1}(n)$ in the tree $T_1$. The embedding includes at least an Euler tour of tree $T_2$ along with the edge $(u, v^*)$. Now if $|T_2| + 2d(u, v^*) \geq (|T_1| + |T_2|)/2$, then the decrease offsets the potential increase. In other words, if $|T_1| - |T_2| \leq 2d(u, v^*)$, then the transformation $B$ does not increase cost. This is certainly true since $v^*$ is a centroid. ■

### 2.3.2 Optimal embeddings are Euler tours

Given any embedding $\pi$ we can apply the transformations $A$ or $B$ to uniterleave the embeddings of the subtrees. Let $v^*$ be the centroid. Let $T_0, T_1, \ldots, T_k$ be the subtrees rooted at $v^*$. Let $|T_i|$ denote the length of an Euler tour of the tree $T_i$. Let the subtrees be arranged in the decreasing order of the lengths of their Euler tours: $|T_0| \geq |T_1| \geq \ldots \geq |T_k|$. Let $\overline{T_0} = T - T_0$.

First we check if the embedding $(T_0 : \overline{T_0})$ has average tree-edge distortion at most that of $\pi$. If so, then we solve the problem recursively on $T_0$ and $\overline{T_0}$.

The other case is when $(T_0 : \overline{T_0})$ has greater average tree-edge distortion than $\pi$. From Lemma 4, we know that neither $\pi^{-1}(1)$ nor $\pi^{-1}(n)$ belongs to $T_0$. Let $\pi^{-1}(1) \in T_{i_1}$, then we can apply transformation $A$ or $B$ to $\pi$ (depending on whether $\pi^{-1}(n) \in T_{i_1}$) and we get the embedding $(T_{i_1} : \overline{T_{i_1}})$. Let the leftmost endpoint of $\overline{T_{i_1}}$ belong to the subtree $T_{i_2}$. Once again we apply the appropriate transformation and get the embedding $(T_{i_1} : (T_{i_2} : T'))$, where $T' = T - T_{i_1} - T_{i_2}$. We continue this process until both endpoints of $T' = T - T_{i_1} - \ldots - T_{i_j}$ belong to $T_0$. At this step, the candidate transformation is $B$. However, it does not decrease cost at this point because $v^*$ is no longer a centroid in $T'$. Hence we must adopt a different line of attack in this case. Let $p$ be the greatest integer such that for all $i \leq p$, and let $e_0$ be the edge from $v^*$ to the root of $T_0$. Then we have

$$2|T_i| \geq (|T_0|) + 2d(e_0) + (|T'|), \tag{2.5}$$

where $T' = T - T_0 - T_1 - \ldots - T_p$. Then we can show that the embedding $(T_1 : T_2 : \ldots : T_p : \overline{T})$, where $\overline{T} = T - T_1 - \ldots - T_p$ has tree-edge distortion smaller than $\pi$. Moreover, since neither $\pi^{-1}(1)$ nor $\pi^{-1}(n)$ belongs to $T_0$, we have $p > 0$.

Thus we have shown that, we can solve the problem recursively on these trees and combine their solutions. From this we get the following important observation.

**Lemma 5** *An optimal non-contracting embedding of a weighted tree $T$ into a line to minimize average tree-edge distortion corresponds to an Eulerian tour.*

### 2.3.3 Algorithm

We describe our recursive algorithm here. Let $T$ be the tree from which the input metric $(V, d)$ arises.

1. Find the centroid $v^*$ of the tree $T$. Let $T_0, \ldots, T_k$ be the subtrees of $T$ rooted at $v^*$.

2. Find the greatest integer $p$ such that for all $i \leq p$, we have $2|T_i| \geq (|T_0|) + 2d(e_0) + (|T'|)$, where $T' = T - T_0 - T_1 - \ldots - T_p$, and $|T_0| \geq |T_1| \geq |T_2| \geq \ldots$.

3. If $p = 0$, then recursively find the embeddings of $T_0$ and $\overline{T_0}$. Output the embedding $(T_0 : \overline{T_0})$.

4. If $p > 0$, then recursively find the embeddings of $T_1, \ldots, T_p, T'$ (where $T' = T - T_1 - \ldots - T_p$). Output the best embedding of these subtrees using the subroutine described below.

**Subroutine:** We now describe the subroutine to combine the embeddings of subtrees $T_1, \ldots, T_i$ rooted at $r$. We want to find the ordering of these subtrees which minimizes

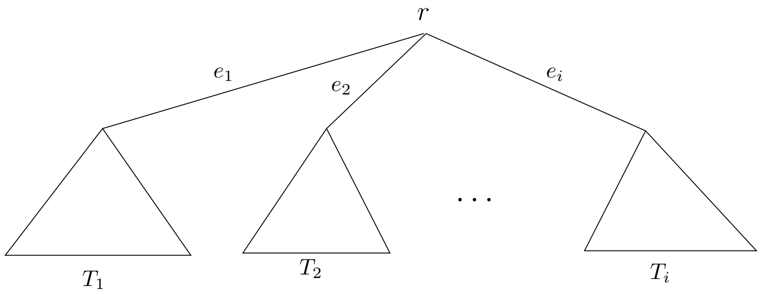


Figure 2.4: Embedding the subtrees

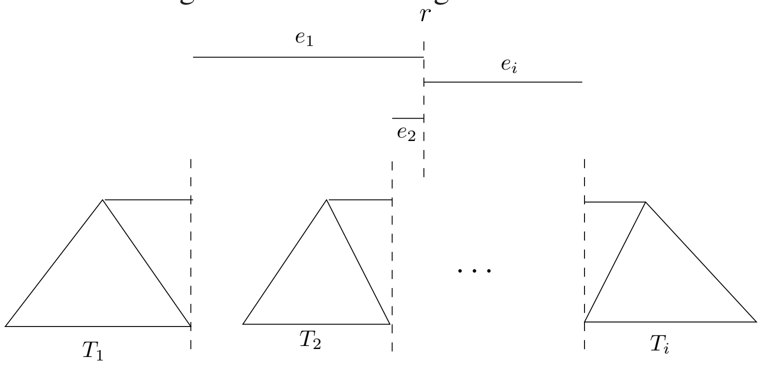


Figure 2.5: Accounting for the lengths of edges

the tree-edge distortion of the embedding. The objective function for this subroutine is the sum of the lengths of edges $e_1, \ldots, e_i$ in the embedding. See Figure 2.5(a). Note that, we only include the part of the edge from $r$ to the closest point of its tree.

Let $d(e_j)$ be the length of the edge $e_j$ in the input metric. Since the embedding is an Eulerian tour, we know that if edge $e_1$ crosses the trees $T_2, T_3$ and $T_4$, e.g., then it is expanded by $|T_2| + |T_3| + |T_4|$. Thus the total length of $e_1$ to account for is $d(e_1) + |T_2| + |T_3| + |T_4| + d'(e_1)$, where $d'(e_1)$ is the part of the length of $e_1$ inside tree $T_1$. The quantity $d'(e_1)$ can be taken as the distance of the root of $T_1$ to its closest endpoint. On the other hand, if there are $j$ edges crossing over the tree $T_q$, then the tree contributes the $|T_q|$ term in the length of each of those edges. Thus, if the tree $T_q$ is $(j+1)^{\text{st}}$ from left or right endpoint, then its contribution to the total cost is $j|T_q| + d(e_q)$.

This suggests that we can find the optimal ordering of the trees using minimum cost matching algorithm. Consider a complete bi-partite graph $K_{i,2i}$ where $i$ is the number of subtrees hanging off the centroid. The $i$ vertices on one side correspond to the trees $T_1, \ldots, T_i$. If the tree $T_q$ is the $(j+1)^{\text{st}}$ from the left in an embedding, this is represented by connecting vertex $q$ on the left side of $K_{i,2i}$ to vertex $j+1$ on the other side, by an edge of weight $2j|T_q| + d(e_q)$. If $T_q$ is the $(j+1)^{\text{st}}$ from the right, then we connect the edge between vertex $q$ to the vertex $i+j+1$ on the other side of the same cost. Finding a minimum-weight matching in this bipartite graph will give us the ordering of trees on left and right side of the root.

**Theorem 6** *There is a polynomial-time algorithm for finding a non-contracting embedding of an input tree metric into a line to minimize average tree-edge distortion.*

We remark that it is not hard to construct instances where the optimal non-interleaving embedding in the same spirit as above provide very poor approximations to the minimum average distortion embeddings even for tree metrics. For example, consider a 3-spider where the vertices are placed at distances $l, l^2, l^3, \ldots$ on each leg. Any non-interleaving embedding has average distortion $\Omega(n)$, whereas the optimal (interleaving) embedding has average distortion $O(1)$.

## 2.4 Discussion

The hardness result for average distortion is not approximation preserving. Therefore, it does not rule out a PTAS for minimizing the average distortion. It is an interesting open question to close this gap.

Lee, Mendel and Naor [Lee et al., 2004] propose a different the notion of average distortion: average taken over the ratios of pairwise distances. It will be interesting to give approximation algorithm for this average distortion.

# Chapter 3

# Additive Distortion

In this chapter, we consider the total additive distortion of the embedding as our objective function. The total additive distortion is the sum of errors in all pairwise distances in the input data. This problem has been shown to be NP-hard by [Saxe, 1979]. We give an $O(\sqrt{\log n})$ approximation for this problem by using Agarwal et al.'s [Agarwal et al., 2005] algorithm for the Min Uncut problem as a subroutine. Our algorithm generalizes to give an $O(\log^{1/2p} n)$ approximation for the $L_p$ norm of the additive distortion.

## 3.1 Problem Formulation

Consider a set of $n$ points, denoted by $[n] = \{1, 2, \ldots, n\}$. The input data consists of an $n \times n$ matrix $D_{n\times n}$. The entry $D_{ij}$ denotes the distance between points $i$ and $j$. We assume that all the entries of $D$ are non-negative and that $D$ is symmetric.[1] Furthermore, we assume that $D_{ii} = 0$ for all $i$.

Let $f : [n] \to \mathbb{R}$ denote a mapping of the input points to the real line. Distance between images of points $i$ and $j$ in the line is given by $f_{ij} = |f(i) - f(j)|$. The total additive distortion (in the $L_1$ norm) is given by

$$L_1(D, f) = \sum_{i,j} |D_{ij} - f_{ij}|.$$

[1]Our results hold even if the input distances in $D_{n\times n}$ do not satisfy triangle inequality, i.e. even if $D$ is not a "metric".

Generalizing this, we can write the $L_p$ norm of the additive distortion as

$$L_p(D, f) = \big(\sum_{i,j} |D_{ij} - f_{ij}|^p\big)^{\frac{1}{p}}.$$

The goal is to find a map $f$ that minimizes the $L_1(D, f)$ (or more generally $L_p(D, f)$).

## 3.2 Approximation for $L_p$ norm

In this section, we give an approximation algorithm for minimizing the $L_p$ norm of the additive distortion.

In Lemma 6, we will show that it is sufficient to look at $r$-restricted mapping of the points into the real line. The problem of finding an optimal $r$-restricted mapping can be cast as a kind of partition problem given the characteristics of the real line.

### 3.2.1 $r$-restricted mappings

Let $r$ be a point in the input. A mapping $f$ of the input points to the real line $\mathbb{R}$ is an $r$-restricted mapping, if distance on the line of all points from $r$ is same as that in the input. Formally, $D_{ri} = |f(r) - f(i)|$ for all $i$.

We will denote an $r$-restricted mapping by $f^r$. We next show that there is always a "good" $r$-restricted mapping. This will enable us to focus only on $r$-restricted mappings which are easier to handle. Agarwala et al. [Agarwala et al., 1999] prove a similar lemma for tree metrics. We adapt their proof for the case of line metrics.

**Lemma 6** *There exists a point* **r** *among the input points such that there is an $r$-restricted mapping $f^r$ that is within a factor of $3$ of the optimal mapping for the $L_p$ norm of additive distortion, for all $p \geq 1$.*

**Proof.** Let $f^*$ denote an optimal mapping of the input points to the line for the $L_p$ norm of additive distortion. We will modify the optimal solution to produce a mapping $f^i$ for each point $i$ in the input. To produce the restricted mapping $f^i$, perturb the distances in $f^*$ so that it becomes $i$-restricted. In particular, if $f^*(j) \leq f^*(i)$ for some $j$, then set $f^i(j) = f^*(i) - D_{ij}$ and if $f^*(j) > f^*(i)$, set $f^i(j) = f^*(i) + D_{ij}$. Our mapping $f^i$ maps point $i$ to $f^*(i)$. It maps rest of the points according to their distance from $i$, while maintaining their order to the left or right of point $i$ in the optimal mapping $f^*$.

Let $\epsilon_{jk}$ denote $|D_{jk} - f^*_{jk}|$. We can write the additive distortion of the optimal mapping as $L_p(D, f^*) = (\sum_{j,k} \epsilon^p_{jk})^{1/p}$. From the construction of the map $f^i$, it follows that $|f^*_{jk} - f^i_{jk}| \le \epsilon_{ij} + \epsilon_{ik}$.

Now we bound the additive distortion of $f^i$ in terms of $\epsilon_{jk}$'s. For all $j, k$ we have,

$$\begin{aligned} |D_{jk} - f^i_{jk}| &\le |f^*_{jk} - f^i_{jk}| + |D_{jk} - f^*_{jk}| \\ &\le (\epsilon_{ij} + \epsilon_{ik}) + \epsilon_{jk} \end{aligned} \tag{3.1}$$

Note that $|x|^p$ is a convex function of $x$ for $p \ge 1$. Therefore, Equation (3.1) gives us the following:

$$\begin{aligned} |D_{jk} - f^i_{jk}|^p &\le (\epsilon_{ij} + \epsilon_{ik} + \epsilon_{jk})^p \\ &\le 3^{p-1}(\epsilon^p_{ij} + \epsilon^p_{ik} + \epsilon^p_{jk}) \end{aligned} \tag{3.2}$$

By an averaging argument, we can say that

$$\min_i \{L_p(D, f^i)^p\} \le \frac{\sum_{i=1}^n L_p(D, f^i)^p}{n}$$

We use Equation (3.2) to bound the sum

$$\begin{aligned} \sum_{i=1}^n L_p(D, f^i)^p &\le \sum_{i=1}^n \sum_{j,k} 3^{p-1}(\epsilon^p_{ij} + \epsilon^p_{ik} + \epsilon^p_{jk}) \\ &\le 3^p n \cdot \sum_{j,k} \epsilon^p_{jk} \\ &= 3^p n \cdot L_p(D, f^*)^p \end{aligned}$$

Therefore, $\min_i L_p(D, f^i) \le 3 \cdot L_p(D, f^*)$ which proves the result. ∎

## 3.3 Algorithm

The result of Lemma 6 proves that it is sufficient to consider $r$-restricted mappings (with a loss of 3 in the approximation factor). Next we describe the algorithm that implements this idea.

**Algorithm A**

1. For each point $r = 1, 2, \ldots, n$, find (approximately) the best $r$-restricted mapping $f^r$.
2. Output a mapping that has the smallest additive distortion among these mappings.

By Lemma 6, the additive distortion of the output of Algorithm A is within a factor of 3 of the optimal additive distortion. As we will show later, finding best $r$-restricted mapping is NP-hard. Therefore, we approximate the optimal $a$-restricted mapping within a factor of $O(\log^{1/p} n)$. From the following observation it follows that the overall approximation factor of our algorithm will be $O(\log^{1/p} n)$.

**Lemma 7** *If $\rho$ is the approximation factor of the algorithm for $r$-restricted mapping, then the solution produced by Algorithm A will be a $3\rho$ approximation for the additive distortion.*

## 3.4 Approximating $r$-restricted mappings

Let $f$ be an $r$-restricted mapping. Without loss of generality, we can assume that $f(r) = 0$. Let $V_1 = \{i \mid f(i) < 0\}$ and $V_2 = \{i \mid f(i) > 0\}$. Note that $[n] = V_1 \cup \{r\} \cup V_2$. Note that, the mapping $f$ is fully characterized by the partition $V_1 \cup V_2$ of $[n] - \{r\}$. Hence, the problem of finding the best $r$-restricted mapping is equivalent to the problem of finding the partition of $V = [n] - \{r\}$ that has minimum additive distortion. Henceforth, we will think of the problem as that of partitioning the input set of points to minimize the *cost* of the partition, i.e. the additive distortion. Here we give an approximation to the $L_p^p$ norm of additive error for the $r$-restricted mapping.

Consider a partition $V_1 \cup V_2$ induced by an $r$-restricted mapping $f$. We can write an expression for its *cost* as follows. Consider two points $x$ and $y$. If they both belong to the same side of the partition, then the contribution of the pair $\{x, y\}$ to the cost of the partition is $c(x, y) = |D_{xy} - f_{xy}|^p = (D_{xy} - |f(x) - f(y)|)^p = |D_{xy} - |D_{rx} - D_{ry}||^p$. On the other hand, if $x$ and $y$ belong to different sides of the partition, then the contribution is $c'(x, y) = |D_{xy} - f_{xy}|^p = |D_{xy} - |f(x) - f(y)||^p = |D_{rx} + D_{ry} - D_{xy}|^p$. Note that $c(x, y)$ and $c'(x, y)$ are completely determined from the input matrix $D_{n\times n}$.

Now, we can think of the problem as a graph partitioning problem where each edge has two costs $c(\cdot)$ and $c'(\cdot)$ associated with it. The $p^{\text{th}}$ power of the cost for the $r$-restricted solution, $L_p(D, f^r)^p$, is same as the objective function for the partition problem. It is given by:

$$\sum_{x,y \text{ on same side}} c(x,y) + \sum_{x,y \text{ on different sides}} c'(x,y). \tag{3.3}$$

### 3.4.1 Two-cost Partition Problem

We are given a complete graph $G = (V, E)$ with two cost functions $c$ and $c'$. We want to find a partition of the vertex set $V = V_1 \cup V_2$ which minimizes $\sum_{i=1,2} \sum_{u,v \in V_i} c(u,v) + \sum_{u \in V_1, v \in V_2} c'(u,v)$.

Note that, if $c(u,v) = 0$ for all $u, v$, then the problem reduces to finding a minimum cut in the graph. On the other hand, if $c'(u,v) = 0$, then the problem is the well known edge deletion for graph bipartition problem (BIP) [Klein et al., 1990]. Our algorithm generalizes the algorithm for graph bipartition given by [Klein et al., 1990, Garg et al., 1996]. The basic idea is to create two copies of each vertex to go on different sides of the partition. To ensure that they are on different sides, we designate each pair as a source-sink pair in the multi-cut subroutine.

**Algorithm B:**

1. Create an auxiliary graph $G'$ from the graph $G$ as follows.

   (a) For each vertex $u$ in the graph $G$, $G'$ has two vertices: $u$ and $u'$.

   (b) For each edge $(u, v)$ we create $4$ edges in $G'$: $(u, v)$, $(u, v')$, $(u', v)$ and $(u', v')$.

   (c) The edges in $G'$ have weights, denoted by $l(\cdot, \cdot)$. Set $l(u, v) = l(u', v') = c(u, v)$ and $l(u, v') = l(u', v) = c'(u, v)$.

2. Use an approximation algorithm for the multi-cut problem (E.g., [Garg et al., 1996]) as a subroutine to find a multi-cut in graph $G'$ with $(u, u')$, for all $u$, as the source-sink pairs. Let $S$ be the set of edges in the multi-cut returned by the subroutine.

3. Construct a set of edges $T$ as follows. If $\{u, v\}$ or $\{u', v'\}$ is chosen in $S$, then include both in $T$. Similarly, if $\{u, v'\}$ or $\{u', v\}$ is chosen, then include both in $T$.

4. Find a bipartition $V_1' \cup V_2'$ of vertices of $G'$ so that $T$ contains all the edges going across the partition.[2]

[2]We will show how to do this in the proof of Proposition 3.1.

5. Output the partition $V_1 \cup V_2$, where $V_i = V_i' \cap V$.

The intuition behind this algorithm is as follows. For the cut represented by $T$, we will show that we can get a partition of vertices in graph $G'$ such that only one of $u$ and $u'$ is in one partition. From the partition of $G'$, we get a bipartition of $G$. The cost of the bipartition of $G$ is related to the cost of multi-cut obtained by above algorithm in the graph $G'$. We prove this in the next lemma.

**Lemma 8** *Algorithm B returns a partition $V' = V_1' \cup V_2'$ of graph $G'$, such that if $u \in V_1'$, then $u' \in V_2'$ and vice versa. Moreover, $\sum_{x \in V_1', y \in V_2'} l(x, y)$ is at most twice that of the multi-cut found after step 2 by Algorithm B separating each $u$ from $u'$.*

**Proof.** Consider the set $S$ of edges found by the multi-cut subroutine whose removal separates each $u$ from $u'$. For each edge $(x, y) \in S$, we also include its "mirror" edge in $T$. i.e. if $(x, y) \in S$, then $(x', y') \in T$ from the graph. Note that, the cost of an edge and its "mirror" edge is same (i.e., $l(x, y) = l(x', y')$). Therefore, the cost of the edges in $T$ is at most twice the cost of edges in $S$.

Now we show that removal of the edges in $T$ breaks the graph in two parts with the desired property. Consider the graph $G' \backslash T$. Construct a graph $H$ whose vertices represent the connected components in $G'$ after removing the edges in $T$. Two vertices $h_1$ and $h_2$ in $H$ are connected to each other if the corresponding connected components in $G'$ have vertices $x$ and $x'$.

In Proposition 3.1, we prove that the graph $H$ is bipartite. Now we can use graph $H$ to construct a partition $V' = V_1' \cup V_2'$ in graph $G'$. Since the vertices in graph $H$ were connected components in graph $G'$, there are no edges crossing the partition $V_1' \cup V_2'$ in graph $G'$. Moreover, bipartiteness of graph $H$ means that each pair of vertices $x$ and $x'$ in graph $G$ is split in the partition. The cost of this partition is at most 2 times the cost of the multi-cut. ■

**Proposition 3.1** *The graph $H$ defined in the proof of Lemma 8 is bipartite.*

**Proof.** For the sake of contradiction, assume that $H$ has a cycle of odd length. Consider three consecutive vertices $u, v$ and $w$ in this odd cycle. Let $v$ be connected to $u$ and $w$.

Let $x$ be a vertex of $G'$ that belongs to the connected component $u$ and defines the edge $\{u, v\}$ in graph $H$. Therefore, $x'$ is the component $v$. Similarly, let $y$

be a vertex in component $w$ and $y'$ be the corresponding vertex in component $v$. Since $x'$ and $y'$ are in the same connected component $v$, there is a path $x' \to y'$ that lies completely inside the component $v$. Since we didn't remove any of the edges on the path $x' \to y'$, all the *mirror* edges haven't been removed either. Therefore the the *mirror* path $x \to y$ connects $x$ and $y$. This contradicts the fact that $x$ and $y$ were in different connected components. This proves that the graph $H$ is a bipartite graph. ■

**Lemma 9** *The cost of the optimal multi-cut is a lower bound on the cost of partition of graph $G$.*

**Proof.** Consider a partition $V = V_1 \cup V_2$ of graph $G$. From this, we can construct a partition of the vertex set of $G'$. Let $V_1' = V_1 \cup \{x' \mid x \in V_2\}$ and $V_2' = V' \backslash V_1'$. Then, removing all the edges in $G'$ crossing this partition ensures that no vertex $x$ is connected to its counterpart $x'$. i.e. The set of edges going across the partition is a multi-cut. The cost of these edges is exactly the cost of the partition of $G$. ■

Recall that GVY algorithm for multi-cut [Garg et al., 1996] is an $O(\log k)$ approximation for $k$ terminals. Here we have $n$ terminals. Therefore by Lemmas 8 and 9, we get an $O(\log n)$ approximation for the best $r$-restricted mapping. Along with Observation 7 give us an $O(\log n)$ approximation for the $L_1$ norm of additive distortion.

## 3.5 Improved algorithm

We show that the two-cost is equivalent to the bipartition problem (BIP).

**Theorem 7** *Two-cost bipartition problem is equivalent to single cost bipartition problem.*

**Proof.** Let the graph $G = (V, E)$. There are two cost functions $c, d : E \to [0, \infty)$. The objective is $\sum_{e+} c(e) + \sum_{e-} d(e)$. Here $e+$ means an edge $e$ which has both endpoints inside a cluster. Similarly, $e-$ means an edge whose endpoints lie in different clusters.

We construct a graph $H = (U, E')$ with a single cost function $x(.)$ as follows. For each $v \in V$, we create two vertices $v, v' \in U$. Let $x(v, v') = \infty$. For each edge $(u, v) \in E$, we create $4$ edges in $E'$ with following costs:

$$x(u, v) = x(u', v') = c(u, v) \text{and}$$

$$x(u', v) = x(u, v') = d(u, v)$$

Now we use a BIP algorithm as black box to solve this problem. Clearly, $v$ and $v'$ cannot be on the same side, as $x(v, v') = \infty$. If $u$ and $v$ are on the same side (and therefore $u'$ and $v'$ on the other side), then the cost to BIP is $x(u, v) + x(u', v') = 2c(u, v)$. If $u$ and $v$ are on different sides in the BIP solution, then $u$ and $v'$ are on one side and $u'$ and $v$ on the other. Thus, the cost is $x(u, v') + x(u', v) = 2d(u, v)$. Thus from the BIP solution, if we drop all the vertices $v'$ we get the desired partition for the two-cost problem. The costs are within constant factor of each other. ■

Now we can use a recent $O(\sqrt{\log n})$-approximation algorithm due to Agarwal et al. [Agarwal et al., 2005] for the two-cost bipartition problem. This improves our approximation factor to $O(\log^1 2pn)$.

## 3.6 Discussion

We can show that the problem of finding the best $r$-restricted mapping is NP-hard by reducing the edge deletion for graph bipartition (BIP) [Garey and Johnson, 1979] to it. Consider a graph $G$. Let $V(G) = n$. We construct a distance matrix $D$ on $n+1$ points $V(G) \cup \{a\}$. Set the diagonal entries $D_{xx}$ to 0. Set $D_{ax} = 1/2$ for all $x \in V(G)$. For all $\{x, y\} \in E(G)$, set $D_{xy} = 1$. Set the rest of the entries to $1/2$. Consider an $r$-restricted mapping. Let $V(G) = V_1 \cup V_2$ be the partition induced by the $r$-restricted mapping. Then the cost of the $r$-restricted mapping is $B(V_1, V_2) + (1/2)(\binom{n}{2} - |E(G)|)$, where $B(V_1, V_2)$ is the number of edges that need to be deleted to obtain $V_1$ and $V_2$ as two sides of a bipartite graph. Therefore, finding the optimal $r$-restricted mapping corresponds to minimizing the number of edges deleted for making the graph $G$ bipartite. This proves that finding the best $r$-restricted mapping is NP-hard. However, this reduction is not approximation preserving. So it does not preclude the possibility of a PTAS for this problem. Getting even a constant factor approximation would be quite interesting.

In the proof of NP-hardness of $r$-restricted mapping problem, we used an input matrix $D$ that does not satisfy the triangle inequality. For input matrix $D$ that is a *metric* (i.e. it satisfies the triangle inequality), it might be possible to get a polynomial time algorithm for the best $r$-restricted mapping.

# Chapter 4

# (Classical) Distortion

In this chapter, we focus on approximating the distortion. The goal is to find an embedding $f : V \to \mathbb{R}$, from the node set $V$ of a graph $G$ into the real line $\mathbb{R}$ that is non-contracting (i.e. $|f(x) - f(y)| \geq d(x,y)$), and has a small *distortion* $D(f) = \max_{x,y \in V} |f(x) - f(y)|/d(x,y)$. We assume that the graph $G$ is an unweighted graph.

**Two Lower Bounds**

We start by giving two basic lower bounds on the distortion of an optimal embedding in terms of structural properties of the input graph $G$.

We call a graph a $k$-*spider* if it can be decomposed into $k$ edge-disjoint simple paths (called *legs*) that share exactly one common node, which is called the *center* of the spider.

**Lemma 10 [3-spider bound]** *Let $G$ be a 3-spider, in which every leg has length at lest $l$. Then any map of $G$ into the line has distortion at lest $2l$.*

**Proof.** Let $c$ denote the center of $G$ and let $x_0, \ldots, x_l$, $y_0, \ldots, y_l$ and $z_0, \ldots, z_l$ denote the first $l+1$ vertices on the three legs of $G$, where counting starts from the center node (i.e. $c = x_0 = y_0 = z_0$). Fix an optimal non-contracting embedding $f^*$, and consider the vertices $x_l, y_l$ and $z_l$ in this embedding.

There must exist vertices $u_l, v_l \in \{x_l, y_l, z_l\}$ such that either $f^*(u_l) \in [f^*(v_l), f^*(c)]$ or $f^*(u_l) \in [f^*(c), f^*(v_l)]$, i.e., the image of $u_l$ lies between $f^*(v_l)$ and $f^*(c)$. Assume w.l.o.g. $f^*(u_l) \in [f^*(c), f^*(v_l)]$, and let $u_i$ and $v_i$ denote the $i$-th node on the path corresponding to $v_l$ and $u_l$, respectively.

Since $f^*(u_l) \in [f^*(c) = f^*(v_0), f^*(v_l)]$, there must exist an index $i$ such that $f^*(u_l) \in [f^*(v_i), f^*(v_{i+1})]$. This gives $|f^*(v_i) - f^*(v_{i+1})| \geq 2l$, because the map

$f^*$ is non-contracting. However, $d(v_i, v_{i+1}) = 1$, which shows that the distortion is at least $2l$, as desired. ■

For the second lower bound we define the following structural property of the graph $G$. The *local density* $\lambda$ of $G$ is defined as

$$\lambda = \max_{v \in V, r \in \mathbb{R}_{>0}} \left\{ \frac{|B(v,r)| - 1}{2r} \right\} ,$$

where $|B(v,r)| = \{u \in V \mid d(u,v) \leq r\}$ denotes the ball of vertices within distance $r$ from $v$. Intuitively, a high local density tells us that there are dense clusters in the graph, which will cause a large distortion. The following lemma formalizes this intuition.

**Lemma 11 [Local Density]** *Let $G$ denote a graph with local density $\lambda$. Then any map of $G$ into the line has distortion at least $\lambda$.*

**Proof.** Fix $v \in V$ and $r \in \mathbb{R}_{\geq 0}$ such that $(|B(v,r)| - 1)/2r$ is maximum, and let $x, y \in B(v,r)$ denote those two nodes from $B(v,r)$ that are farthest apart in the optimum map $f^*$. Assume, $f^*(x) < f^*(y)$.

All vertices from the ball $B(v,r)$ are mapped to a point from the interval $[f^*(x), f^*(y)]$. There are $|B(v,r)|$ such vertices, and since the embedding $f^*$ is non-contracting we get $|f^*(x) - f^*(y)| \geq (|B(v,r)| - 1)$. However, the shortest-path distance between $x$ and $y$ in $G$ is at most $d(x,y) \leq d(x,v) + d(v,y) \leq 2r$. Hence, the distortion is at least $(|B(v,r)| - 1)/2r$. ■

A simple corollary of this lower bound is the following:

**Corollary 4.1 (Size of $r$-ball)** *Let $D$ be the optimal distortion for embedding the graph $G$ into the line. Then $|B(v,r)| = O(r \cdot D)$, for every ball $B(v,r)$.*

The local density lower bound has been widely studied in the context of the *Minimum Bandwidth Problem*, where it is known that the minimum bandwidth is within a polylogarithmic factor of the local density. However, for our problem this bound can be rather weak. For example, in the case of a 3-spider with legs of length $n/3$, the local density is $3/2$, whereas the optimum distortion is at least $\Omega(2n/3)$ by Lemma 10. This shows that a combination of the local density bound and the 3-spider bound is needed in order to obtain a reasonable approximation ratio for our embedding problem.

The following result will be useful for our algorithms:

**Theorem 8 (Matoušek [Matoušek, 1990])** *Any $n$-point metric can be embedded into the real line with distortion at most $O(n)$.*

For the case of unweighted graphs (which is all we need), this can be proved by just laying the vertices out in any depth-first order – the distortion of such a layout is at most $(n-1)$. (We will show this in Lemma 13.)

## 4.1 $O(\sqrt{n})$-Approximation algorithm for general graphs

Before describing the embedding algorithm, we give the basic idea behind it. If the optimal distortion to embed $G$ into the line is $D$, we show that $G$ has a so-called "diametric path" such that all the vertices of $G$ are "close to" (i.e., within distance $D$ of) this path. This allows us to cut this diametric path at every $D$ steps and extend this partitioning to the rest of the graph, and hence get pieces of diameter at most $D$. We show how to embed each of these components individually, and finally how to stitch these embeddings together to get an embedding of the graph $G$.

**Lemma 12 [Concentration around diametric path]** *If $s$ and $t$ are two vertices in the graph $G$ such that $\mathrm{d}(s,t)$ is maximum, then the distance of all other points from the shortest $s$-$t$ path is at most $D$.*

**Proof.** Let $\mathrm{path}(s,t)$ denote the shortest $s$-$t$ path. Assume for contradiction that there is a vertex $x$ whose distance from $\mathrm{path}(s,t)$ is larger than $D$, and let $c$ denote the vertex on $\mathrm{path}(s,t)$ that is closest to $x$.

By the definition of $x$, we have $d(c,x) > D$. Moreover, since $(s,t)$ is the pair with maximum distance $d(s,t)$, the inequalities $d(c,s) > D$ and $d(c,t) > D$ hold. Hence, the union of $\mathrm{path}(s,t)$ and the shortest $c$-$x$ path forms a 3-spider $S$ in which every leg has length at least $D+1$.

Lemma 10 implies that the distortion for embedding the 3-spider $S$ is at least $2(D+1)$. Since, for every $x,y \in S$, the distance between $x$ and $y$ in $S$ and the distance between $x$ and $y$ in $G$ differs at most by a factor of 2, the distortion for embedding $G$ into the line is at least $D+1$. This gives a contradiction. ■

Now we partition the node set of $V$ into small pieces. Assume for simplicity that the length $d(s,t)$ of the diametric path is a multiple of $D$. Let $s = v_0, v_1, \ldots, v_k = t$ denote the vertices on $\mathrm{path}(s,t)$ such that $d(v_i, v_{i+1}) = D$. We assign each node in $G$ to the node $v_i$ that is nearest to it (ties are broken arbitrarily), and form a component $X_i \subset V$ from all

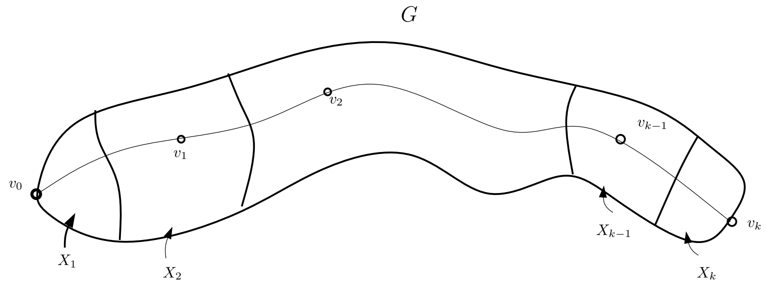


Figure 4.1: Partition into small balls

nodes assigned to vertex $v_i$. The following claim shows that the components $X_i$ created by this process have a low diameter.

**Claim 4.2** $X_i \subset B(v_i, \frac{3}{2}D)$.

**Proof.** It follows from Lemma 11 that for any node $u$ there is a node $c$ on the path $\mathrm{path}(s,t)$ with $d(u,c) \leq D$. Since, the vertices $v_i$ are placed in distance $D$ on this path, there must exist an index $i$ for which $d(c, v_i) \leq D/2$. This shows that for each node $u$ there is a node $v_i$ with $d(u, v_i) \leq \frac{3}{2}D$, and hence $X_i \subset B(v_i, \frac{3}{2}D)$. ■

**Lemma 13** *For each component $X_i$, there is a non-contracting embedding into the interval $[0, L]$, where $L = O(D^2)$.*

**Proof.** A component $X_i$ is contained in the ball $B(v_i, \frac{3}{2}D)$. A depth-first search (DFS) on this ball takes at most $O(D^2)$ steps since there are at most $O(D^2)$ nodes in the ball, due to Corollary 4.1. We map a node in $X_i$ that is visited in the $k^{\mathrm{th}}$ step of the DFS to $k$. This gives a non-contracting mapping into the interval $[0, L]$, where $L = O(D^2)$ is the maximum number of steps needed by the DFS. ■

**Theorem 9** *Let $f_i : X_i \to [0, L]$ denote the embedding of $X_i$ in Lemma 13, where $L = O(D^2)$. The embedding $f : V \to \mathbb{R}$ that maps a node $v \in X_i$ to $i \cdot L + f_i(v)$ has distortion at most $O(D^2)$.*

**Proof.** Clearly, the distortion of an edge between two nodes from the same component is at most $L = O(D^2)$, because both nodes are mapped to the same interval.

Consider an edge $\{x, y\}$, where $x \in X_i$ and $y \in X_j$ for $i \neq j$. First, we note that $|i - j|$ cannot be bigger than $4$; otherwise the path from $v_i$ to $v_j$ that goes via the $\{x, y\}$ edge is shorter than the diametric shortest path we started with. Hence, we have $|i - j| < 4$. In this case $x$ and $y$ are mapped into an interval of length at most $4L = 4D^2$, which implies a distortion of $O(D^2)$. This completes the proof of the theorem. ∎

The following argument turns the above result into a $O(\sqrt{n})$-approximation algorithm.

**Theorem 10** *There exists a polynomial time $O(\sqrt{n})$-approximation algorithm for embedding an unweighted graph into a line to minimize the distortion.*

**Proof.** Assume that we know the value of the optimal distortion $D \in [1, n]$; if $D \geq \sqrt{n}$, then we just output a DFS tour of the entire graph. Since the length of the DFS tour is $O(n)$, the distortion is $O(n) \leq O(\sqrt{n}) \times D$, giving us the claimed embedding.

On the other hand, if $D < \sqrt{n}$, then we use the algorithm from Theorem 9 find an embedding with a distortion of $O(D^2) \leq O(\sqrt{n}) \times D$; this completes the proof. ∎

## 4.2 Better embeddings for unweighted trees

For the case of trees, we use a similar framework as for graphs: we divide the tree along the "diametric path" and obtain connected components $X_1, \ldots, X_k$ with each $\mathrm{diam}(X_i) \leq D$ and $|X_i| = O(D^2)$. Instead of taking the depth-first tour to embed each $X_i$ as in Lemma 13, we give a more sophisticated embedding.

### 4.2.1 Prefix Embeddings

We first prove that it suffices to consider embeddings where each prefix of the associated tour forms a connected component of the tree; this will allow us to considerably simplify all our later arguments.

**Lemma 14 [Prefix Embeddings]** *Given any graph $G$, there exists an embedding of $G$ into the real line with the following two properties:*

1. *Walk from left to right on the line, the set of points encountered up to a certain point forms a connected component of $G$.*

2. *The distortion of this map is at most twice the optimal distortion.*

**Proof.** Consider the optimal embedding $f^*$, and let $v_1, v_2, \ldots, v_n$ be the order of the points in this embedding. (We will blur the distinction between a vertex $v$ and its image $f^*(v)$ on the line.) Without loss of generality, we can assume that the distance between any two adjacent points $v_i$ and $v_{i+1}$ in this embedding is their shortest path distance $d(v_i, v_{i+1})$.

Let $i$ be the smallest index such that $\{v_1, v_2, \ldots, v_i\}$ does not form a connected subgraph; hence there exists some vertex on every shortest $v_{i-1}$-$v_i$ path that has not yet been output. We pick one of these shortest paths $P$, take the vertex $w$ in $P - \{v_1, v_2, \ldots, v_{i-1}\}$ closest to $v_{i-1}$, and place it at distance $d(v_{i-1}, w)$ to the right of $v_{i-1}$ in the embedding. We repeat this process until Property 1 is satisfied; it remains to bound the distortion we have introduced.

Note that the above process moves each vertex at most once, and then it is moved to the left. We claim that each vertex is moved by at most a distance $D$, where $D$ is the optimal distortion. Indeed, consider a vertex $w$ that was moved when addressing the $v_{i-1}$-$v_i$ path, and let $v_k$ be a neighbor of $w$ among $v_1, \ldots, v_{i-1}$. Note the distance $|f^*(v_k) - f^*(w)|$ between these two vertices is at most $D$ in the optimal embedding. Since $w$ stays to the right of $v_k$, the distance by which $w$ was moved is at most $D$.

In short, though the above alterations moved vertices to the left, whilst keeping others at their original locations in $f^*$, the distance between the endpoints of an edge increased by at most $D$. Since the distance $|f^*(v) - f^*(u)|$ was at most $D$ to begin with, we end up with an embedding with distortion at most $2D$, proving the lemma. ∎

Henceforth, we will only consider embeddings that satisfy the properties stated in Lemma 14. The bound on the increase in distortion is asymptotically the best possible: for the case of the $n$-vertex star $K_{1,n-1}$, the optimal distortion is $\approx n/2$, but any prefix embedding gets a distortion of at least $n - 2$.

### 4.2.2 The Embedding Algorithm

In this section, we give an algorithm which embeds trees with distortion $g(D) = O(\lambda\sqrt{D \log D} + D)$, where $\lambda$ is the local density and $D$ the optimal distortion. The algorithm proceeds in rounds: in round $i$, we lay down a set $Z_i$ with about $g(D)$ vertices. To ensure that the neighbors of vertices are not placed too far away from them, we enforce the condition that the vertices in $Z_i$ include all the neighbors of vertices in $\cup_{j<i} Z_j$ that have not already been laid out.

It is this very tension between needing to lay out a lot of vertices and needing to ensure their neighbors do not hurt us later, that leads to the following algorithm. In fact, we will mentally separate the action of laying out the neighbors of previously embedded vertices (which we call the *BFS part* of the round) from that of laying out of new vertices (which we call the *DFS part*).

We assume that we know the left-most vertex $r$ in the prefix embedding; we can just run over all the possible values of $r$ to handle this assumption. Let $N(X)$ denote the set of neighbors of vertices in the set $X \subseteq V$.

We define a *light path ordering* on the vertices of the tree $T$. The light path ordering is a DFS ordering which starts at root $r$ and at each point enters the subtree with smallest number of points in it.

**Algorithm** Tree-Embed**:**

1. let $C \leftarrow \{r\}$ denote the set of vertices already visited. Set $i \leftarrow 1$.
2. while $C \neq V(T)$ do
   **(Round $i$ BFS)**
3. Visit all vertices in $N(C) \setminus C$; let $C \leftarrow C \cup N(C)$
   **(Round $i$ DFS)**
5. set $B$ to be a set of $g(D)$ vertices of $V(T) \setminus C$ in the *light path ordering*.
   Visit all vertices in $B$; let $C \leftarrow C \cup B$.
6. endwhile

**Lemma 15 [Number of rounds]** *The algorithm* Tree-Embed *requires* $O(\sqrt{D \log^{-1} D})$ *iterations to complete.*

**Proof.** By the very definition of the algorithm, the set $C$ grows by at least $g(D)$ in every iteration. Note that the diameter of the tree is bounded by $O(D)$ and its local density is $\lambda$. Therefore, the number of points in the tree is $O(\lambda D)$. Hence, within $O(\lambda D/g(D)) = O(\sqrt{D \log^{-1} D})$ rounds, all the vertices of the tree will be visited. ■

The heart of the proof is showing that visiting the vertices in Steps 3 and 5 does not incur too much distortion; it may be the case that the size of $N(C) \setminus C$ may be too large, or even that these vertices may be separated very far from each other.

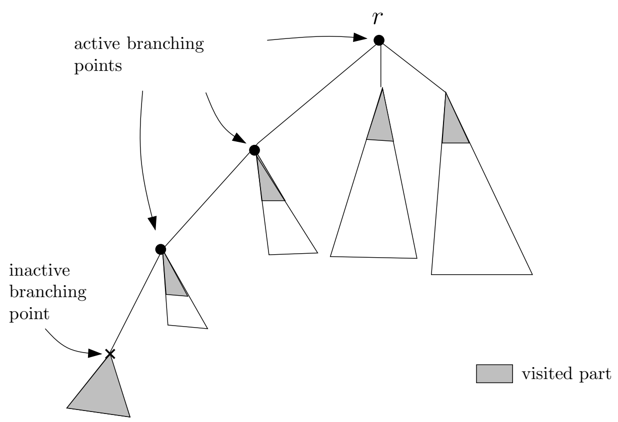


Figure 4.2: A typical snapshot of Algorithm Tree-Embed

**Lemma 16 [Span of boundary]** *The size of the induced spanning tree on the boundary $N(C) \setminus C$ is bounded by $g(D)$.*

**Proof.** Consider the set $C_i$ of vertices that have been visited by round $i$. Consider a vertex $x$ visited in round $j$ of the DFS for some $j \leq i$. Note that the children of the vertex $x$ will be visited *after* $x$. We say that $x$ is a *branching point* if not all the children of $x$ were visited in the same round as $x$. The branching point $x$ is *active* after round $i$ if at least one of the vertices below it has not been visited by round $i$; otherwise it is *inactive*. We claim that all the active branching points in $C_i$ lie on some root-leaf path. This follows because the light path ordering is a DFS ordering. Therefore, if some vertices below a branching point $x$ have not been visited, then the DFS part of the algorithm will not visit a different subtree.

Note that each active branching point (except possibly the lowest one) has at least two children and the algorithm visits the child which has a smaller number of vertices in its subtree. Therefore, the number of active branching points on a root to leaf path is at most $O(\log D)$.

We claim that every point in $N(C_i) \setminus C_i$ is within a distance of $i + 1$ of some active branching point. We prove this by induction on $i$. Before the first round, this property is true, since $C_0 = \{r\}$. Now assume the property for $i - 1$ and consider a vertex $v \in N(C_i) \setminus C_i$. Let $u$ be the neighbor of $v$ such that $u \in C_i$. If $u$ was visited in the round $i$ of the DFS, then $u$ is an active branching point, since its child $v$ has not been visited in the same round. Otherwise, if $u$ was visited in round $i$ of the BFS, then $u$ is within distance $i$ of some branching point $x$. Since $v$ is below $x$ and has not been visited after round $i$, the branching point $x$ must be active. Therefore,

$v$ is within distance $i+1$ from some active branching point.

Let $x$ be an active branching point and let $N_x$ contain the points from $N(C_i)\setminus C_i$ that are within distance $i+1$ from $x$. Then, we can bound the span of the induced tree on $N_x$ using the local density bound. The number of vertices in the induced tree on $N_x$ is bounded by $(i+1)\lambda$. Therefore, the sum of spans over all the active branching points is at most $O(\lambda\sqrt{D \log D})$. Note that, all the active branching points are on a single root-leaf path. Therefore, connecting all the branching points in $N(C_i)\setminus C_i$ requires only a path of length $O(D)$. Hence, the total span of vertices in $N(C_i)\setminus C_i$ is bounded by $g(D)$. ■

**Lemma 17** *The span of the tree induced on the vertices visited in any iteration is bounded by $g(D)$.*

**Proof.** From Lemma 16, the span of the vertices visited in Step 3 of the algorithm is bounded by $O(\lambda\sqrt{D \log D}+D)$. The number of new vertices visited in Step 5 of the algorithm is bounded by $g(D)$. Since, we visit a set of connected components, their span is bounded by $g(D) + \mathrm{span}(N(C) \setminus C)$. Therefore, the span of the vertices visited in each iteration is bounded by $O(\lambda\sqrt{D \log D} + D)$. ■

**Lemma 18** *The distortion of the embedding produced by Algorithm* Tree-Embed *is $g(D) = O(\lambda\sqrt{D \log D} + D)$.*

**Proof.** For a pair of vertices that are visited during the same iteration, the distance in the embedding is bounded by $g(D)$ (from Lemma 17). Therefore, the distortion of such a pair is bounded by $X$. So, consider an edge $(x, y)$ such that $x$ and $y$ were visited in different iterations. Note that, step 1 of the algorithm ensures that if $x$ was visited in iteration $i$, then $y$ was visited in iteration $i+1$. Therefore, the distance between $x$ and $y$ in the embedding is bounded by $g(D)$. Hence, the distortion is bounded by $g(D) = O(\lambda\sqrt{D \log D} + D)$. ■

**Concatenating the embeddings** In order to concatenate the embeddings of $X_1, X_2, \ldots$, it is enough to observe that since the input graph is a tree, there is only one edge connecting components $X_i$ and $X_{i+1}$ for all $i$. Let $r_i$ denote the root of $X_i$ and $s_i$ denote the vertex in $X_i$ connecting to $r_{i+1}$, the root of $X_{i+1}$. It follows from our decomposition that $d(r_i, s_i) \leq O(D)$. To produce an embedding of the component $X_i$ using Algorithm Tree-Embed, we use a light path ordering of $X_i$ assuming that the subtree containing $s_i$ is heaviest subtree.

Hence $s_i$ is last in the light path ordering of $X_i$ and is visited last by Algorithm Tree-Embed. This makes sure that the distortion of the edge $(s_i, r_{i+1})$ is also within the bound. Changing the light path ordering in this way, does not affect the bound on distortion proved in Lemma 18. Thus we get the following result.

**Theorem 11** *There is a polynomial time algorithm that finds an embedding of an unweighted tree with distortion $O(\lambda\sqrt{D \log D} + D)$.*

**Corollary 4.3** *There is a polynomial time algorithm that finds an embedding of an unweighted tree with distortion within a factor $O((n \log n)^{1/3})$ of the optimal distortion.*

## 4.3 Hardness results

**Theorem 12** *The problem of minimizing the distortion of embedding an unweighted graph into line is NP-hard.*

**Proof.** The reduction is from the Hamilton Path problem. Suppose we are given a graph $H = (V', E')$ with $|V'| = h$, we create a graph $G$ thus: we take two copies of $H$ and a new vertex $r$, and add edges from $r$ to all the vertices in both copies of $H$. We also set the weight of all edges to be 1. We now ask whether $G$ is embeddable with distortion $h$ into the real line.

Clearly, we can assume that the embedding is an expansion, since scaling does not change the distortion. If $H$ had a Hamilton path $(v_1, \ldots, v_h)$, we can map $r$ to the origin, and the two copies of $v_i$ to the points $+i$ and $-i$ respectively. This can be easily checked be an expansion, and to have distortion exactly $h$. Thus accepting instances get mapped to accepting instances.

On the other hand, if $H$ has no Hamilton path, then let us look at any embedding $\phi$. By the Pigeon-Hole principle, $r$ will have at least $h$ vertices to one side, and at least one consecutive pair of these vertices will have distance 2 between them (since they will not be connected: either they belong to different copies of $H$, or they don't have an edge between them) while all the others have distance at least 1. Thus the distortion of the edge connecting $r$ to the furthest vertex on this side will be at least $h + 1$, thus completing the proof. ■

MAX-SNP hardness of the problem follows similarly, by an easy reduction from metric-TSP with edge weights 1 and 2.

We now show that embedding a (weighted) tree into a line is NP-hard.

**Theorem 13** *The problem of minimizing the distortion of embedding a (weighted) tree into a line metric is NP-hard.*

**Proof.** We will reduce from the set partitioning problem (PARTITION), which is NP-complete [Garey and Johnson, 1979] (but only weakly so). In PARTITION, the input has $n$ positive integers $a_1, a_2, \cdots, a_n$ with $\sum_i a_i = 2L$ and we want to decide whether there is some subset of these numbers which add up to exactly $L$.

Given an instance of this problem, let us construct a star with edge lengths as described below. Let $n$ of the edges correspond to the input for PARTITION and hence have lengths $a_i$. We call these the *short edges*. We also add $2(n+1)$ *long edges*, each with length $3L$. We now ask whether $T$ can be embedded onto the real line with distortion at most $D = (2n + 5/3)$. In the following discussion, we always assume that the root is embedded at the origin.

If we have a positive instance of PARTITION, without loss of generality assume that the first $k$ numbers add up to $L$ and that $a_1 \leq a_2 \leq \ldots \leq a_k$, and $a_{k+1} \leq a_{k+2} \leq \ldots \leq a_n$. We embed the vertices corresponding to the first $k$ short edges on the positive half of the axis with

$$\phi(v_i) = a_i + \sum_{0<j<i-1} 2a_j,$$

and the remaining vertices at the positions

$$\phi(v_{k+i}) = -\, a_{k+i} - \sum_{0<j<i-1} 2a_{k+j}.$$

Now the vertices corresponding to the long edges can be placed at positions $\pm(2L + (2i+1)3L)$, for $0 \leq i \leq n$. It can be easily checked that this embedding is an expansion and furthermore, the distortion is at most $(6n+5)/3 = 2n + 5/3 = D$.

On the other hand, take a negative instance of PARTITION. Consider the optimal embedding of this graph which is an expansion, and look at the vertices corresponding to the long edges. Clearly, no more than $(n+1)$ of these can be placed on one side, else the distortion for the last vertex would be at least $[2(n+1)+1] \geq 2n+3$. Hence we have exactly half these vertices on either side in an optimal embedding of such an instance. Since this is a negative instance of PARTITION, any partition of the vertices of the short edges will cause the final vertex to be at distance at least $2(L+1)+(2n+1)3L$, and hence we have distortion at least $2n+5/3+2/3L > D$. ∎

## 4.4 Improved Embedding

In this section, we consider a special case of the problem where the input tree $T$ has subtrees with size and depth $\leq D$. In this case, we are able to give $O(\log D)$-approximation to the distortion.

## 4.5 Lower bounds:

We prove the following two lower bounds on the distortion based on the structure of the tree. These arguments are based on the work done per round and the span of the boundary.

**Lemma 19 [Small Trees Bound]** *Let all the subtrees of the tree $T$ have size smaller than $s$. Let $V$ denote the total number of vertices in those subtrees. Then the distortion of any embedding of $T$ is at least $V/\sqrt{s}$.*

**Proof.** If there are at least $V/2$ vertices within a distance of $\sqrt{s}$ from the root, then the local density of the tree $\Delta \geq V/(2\sqrt{s})$. Therefore, the distortion is $\Omega(V/\sqrt{s})$.

So now assume that more than $V/2$ vertices are at a distance of $\sqrt{s}$ or more from the root. Consider the situation after round $\sqrt{s}/2$ of the optimum embedding. In $\sqrt{s}/2$ rounds the optimum embedding can visit at most $D \cdot \sqrt{s}/2$ vertices out of the $V/2$ vertices that are 'far' from the root. Thus, there are at least $\frac{V}{2} - \frac{D\sqrt{s}}{2}$ vertices that are at least $\sqrt{s}$ far from root and not yet visited. Since each subtree has size at most $s$, there must be at least $\frac{V}{2s} - \frac{D}{2\sqrt{s}}$ subtrees that have vertices yet to be visited. Each such subtree contributes at least $\sqrt{s}$ to the span of the boundary. Hence the total span of the boundary at this stage is $\Omega(\frac{V}{2\sqrt{s}} - \frac{D}{2})$. Consequently, the distortion is $\Omega(\frac{V}{\sqrt{s}})$. ■

**Lemma 20 [Large Trees Bound]** *Let the tree $T$ have $k$ subtrees of size at least $s$ each. Then the distortion of any embedding of $T$ is at least $k\sqrt{s}$.*

**Proof.** The total number of vertices in $T$ is at least $ks$. Suppose at least $k/2$ subtrees have depth smaller than $\sqrt{s}$. Then at least $ks/2$ vertices out of the total are within a distance of $\sqrt{s}$ from the root. Hence, the local density of the tree $\Delta \geq k\sqrt{s}/2$ and the distortion is $\Omega(k\sqrt{s})$.

Now suppose that more than $k/2$ trees have depth more than $\sqrt{s}$. Consider the the optimum embedding after $\sqrt{s}/2$ rounds. Up to this point, the optimum embedding could have visited at most $D\sqrt{s}/2$ vertices. Thus the optimum embedding could have finished at most $D/(2\sqrt{s})$ subtrees by this point in time. Therefore, at least $k/2 - D/(2\sqrt{s})$ trees with depth $\sqrt{s}$. Therefore, the span of the boundary is at least $k\sqrt{s}/2 - D/2$. Hence the distortion is $\Omega(k\sqrt{s})$. ■

## 4.6 Algorithm

Next we describe our randomized algorithm and give a rough sketch of its analysis.

---

**1:** For each subtree, pick a delay $d$ u.a.r. from $[1, 2)$

**2:** In the BFS rounds, we visit each subtree with its delay $d$, i.e. if the delay is $y/x$, then visit the subtree $x$ times in $y$ rounds.

**3:** In each DFS round, find the smallest subtrees hanging off the boundary that can be visited in $O(D \log n)$ work and visit them.

---

Figure 4.3: Algorithm Random-Delay

**Observation 4.4** *The BFS part visits each subtree at least once in every two consecutive rounds.*

This follows because the delay is at most 2. Furthermore, the observation implies that the work done in the DFS between two BFS visits to a subtree is at most $O(Dlogn)$.

## 4.7 Analysis

The main idea is to show that by the end of round $t$, the algorithm $R$ manages to finish all the subtrees hanging off the boundary with size up to $t^2$.

Let $r$ denote the root of the subtree. For each vertex $v$, define $T_v$ = the set of the vertices $w$ such that the $r \to w$ path contains the vertex $v$. The set $T_v$ is the subtree rooted at $v$. Let $|T_v|$ denote the number of vertices in $T_v$. In particular, we have $|T_v| = 1$ iff $v$ is a leaf.

We define the set of subtrees of size less than $s$ and rooted at no more than $t$ from the root as follows.

$$V_s^t = \{u \in T_v \mid v \in B(r,t) \text{ and } |T_v| \leq s\},$$

where $B(r,t)$ is the ball of radius $t$ around $r$.

**Lemma 21** *Consider the set $V_s^t$. If $s \leq c \cdot t^2$, then the number of vertices in this set is bounded by $2c\, D \cdot t$.*

**Proof.** Consider the subtrees of size smaller than $s$ that are rooted within distance $t$ from the root. Let $N$ denote the number of vertices in these subtrees. After $t$ rounds, the optimum algorithm visits at most $D \cdot t$ vertices. Moreover, all the subtrees remaining after $t$ rounds have size smaller than $ct^2$. The number of vertices remaining is at least $N - Dt$. Therefore, it follows from Lemma 19 that the distortion must be at least $(N - Dt)/(t\sqrt{c})$. Hence $N \leq 2c'\, D \cdot t$. ■

## 4.8 A different view of the algorithm

Randomize the tree. Run the greedy algorithm on it. In other words, visit smallest subtrees in each round with a budget of $O(D \log D)$. We claim that the optimum distortion on randomized tree is within a constant factor of the optimum distortion on the original tree.

Now consider the randomized tree, i.e. each of the subtrees is stretched by a random factor between $[1, 2]$. In the rest of the analysis, we focus solely on the randomized tree.

**Lemma 22** *Consider the set $V_s^t$ in the randomized tree. If $s \leq c \cdot t^2$, then the number of vertices in this set is bounded by $2c\, D \cdot t$.*

**Proof.** This follows directly from Lemma 21. The randomization only stretches the trees, and hence $|V_s^t|$ can only go down. ■

**Lemma 23** *The probability that vertex $v \in T$ lands at level t in the randomized tree is bounded by $2/t$.*

**Proof.** Let the vertex $v$ be at a distance $t'$ from the root in the original tree. The randomization process maps it in the interval $[t', 2t']$ uniformly. Therefore, for $v$ to get mapped at distance $t$ from the root (i.e. level $t$), it must be the case that $t' \leq t \leq 2t'$. If this condition holds, then the probability that $v$ will get mapped to $t$ is $\frac{1}{t'} \leq \frac{2}{t}$. This proves the lemma. ■

Consider a level $t$. Define $W_t$,the "new work created" at a level $t$, as

$$W_t = \{u \in T_v \mid v \text{ is at level } t \text{ and } |T_v| \leq t^2\}$$

In the next lemma we bound the quantity $W_t$.

**Lemma 24** *The new work created in round $t$ is bounded by $O(D \log D)$ with high probability.*

**Proof.** Suppose there are $k$ subtrees $T_1, T_2, \ldots, T_k$ at the root. Let $X_i$ be the number of vertices from the subtree $T_i$ in the new work created at level $t$. Thus $X_i$ is a random variable whose maximum value is $D$. The new work created in round $t$ is given by

$$W_t = \sum_i X_i$$

We first compute the expected value of $W_t$. Note that only the vertices from $V_{t^2}^t$ can be included in $W_t$. Moreover, we will count a subtree $T_v$ as new work created in round $t$ iff the subtree from the parent of $v$ has size bigger than $t^2$.

Thus a vertex $w \in T_v$ is in $W_t$ if $v$ is at level $t$ in the randomized tree. Hence, probability that a subtree $T_v$ from $V_{t^2}^t$ gets counted in $W_t$ is bounded by $2/t$. Therefore, we can bound the expected size of $W_t$ as follows.

$$\mathbf{E}[W_t] \leq \sum_v |T_v| \cdot \mathbf{Pr}[v \text{ at level } t] \leq \frac{2}{t} 2Dt \leq 4D$$

In the previous equation, we bounded the size of $T_v$ using Lemma 21.

Now using a Chernoff bound, we get

$$\mathbf{Pr}[W_t \geq 24D \log D] \leq \exp(-2 \log D)$$

This proves the lemma. ∎

Intuitively, $W_t$ is the amount of new work created as our BFS boundary reaches level $t$. Since $W_t$ is bounded by $O(D \log D)$, our algorithm is able to finish the new work created.[1]

We also need to consider (long skinny) subtrees of size roughly $t^2$ that are rooted at $t' \ll t$. We haven't considered such subtrees as part of "new work created" anywhere, since around level $t'$ they were much bigger than $t'^2$ and around level $t$, they weren't new anymore. To handle this, we prove the following lemma with a stronger invariant.

**Lemma 25** *By round $2^i$, the algorithm Random-Delay visits all the subtrees of size $2^{2i+2}$ rooted at $2^i$ or lower.*

[1] We also need to bound the span of the new work created, since a whole bunch of such subtrees could be created.

**Proof.** Consider the subtrees of size up to $2^{2i+2}$ that are rooted at $2^i$ or lower. From Lemma 22, it follows that the number of vertices in such trees is bounded by $D \cdot 2^{i+3}$. We will refer to this as the *old work*.

Consider the rounds from $2^{i-1}$ to $2^i$. In each of these rounds, only $O(D \log D)$ new work is created in subtrees of size up to $2^{2i+2}$ (from Lemma 24). Therefore, algorithm R is able to finish all the new work in the same round in which it was created. Moreover, algorithm Random-Delay does at least $16D$ amount of *old work* in each such round.

Therefore, at the end of round $2^i$, the algorithm has visited all subtrees of size up to $2^{2i+2}$ that are rooted at $2^i$ or lower. ∎

Thus we maintain the invariant that after $t$ rounds, only subtrees of size bigger than $t^2$ are hanging off the boundary. (In fact, we have proven a stronger invariant, but this one suffices for the rest of the proof.) The following lemma proves that number of such trees is small.

**Lemma 26** *Number of subtrees of size $\geq t^2$ rooted at depth t is at most $O(D/t)$.*

**Proof.** Let $k$ be the number of trees of size bigger than $t^2$. Call a subtree half-visited, at least half of its vertices have been visited. By round $t$, at most $D \cdot t$ vertices can be visited by any algorithm. In other words, at most $2D/t$ of these subtrees can be half-visited by any algorithm. Therefore, at least $(k - 2D/t)$ subtrees each with $\geq t^2/2$ vertices would still remain. Using Lemma 20, we see that the distortion is at least $(k - \frac{2D}{t}) \cdot \frac{t}{2}$. Therefore, $k \leq \frac{4D}{t}$ which proves the lemma. ∎

**Lemma 27** *The span of the boundary after round t is bounded by $O(D)$ for any t.*

**Proof.** Since the number of subtrees remaining after $t$ rounds is bounded by $O(D/t)$, the span of the boundary after $t$ rounds is bounded by $O(D)$. ∎

Since the work in each round is bounded by $O(D \log D)$ and the span is bounded by $O(D)$ after each round, the distortion of the algorithm R is bounded by $O(D \log D)$. Thus we get the following theorem.

**Theorem 14** *The Algorithm Random-Delay is a polynomial-time (randomized) algorithm that computes an embedding of the input tree to the line with distortion $O(D \log D)$, where $D$ is the optimal distortion.*

## 4.9 Discussion

The main open question is whether there is a $O(\log n)$ approximation for the case of unweighted trees. Our algorithm does not extend to the case when subtrees could have size larger than $D$.

As an alternate approach, we can write the following linear programming formulation for the distortion problem. We use the following properties:

1. There are at most $D$ rounds. The diameter of the tree is $D$, therefore any two points in the tree should be embedded within a distance $D^2$ of each other, i.e. within $D$ rounds.

2. If a vertex $v$ is visited in round $t$, then all of its children must be visited by round $t+1$.

3. A vertex is *on the boundary* after round $t$, if its parent has been visited in round $t$ but the vertex itself was not. The span of the vertices on the boundary is the size of the Steiner tree connecting these vertices. The span of the vertices on the boundary after any round $t$ plus the number of new vertices visited in round $t$ is bounded by $D$.

To formulate the LP, we have variables $x_t(v)$ for each round $t$ and each vertex $v$. It is zero if the vertex $v$ has not been visited by round $t$ and is 1 from the round $t$ it is visited. We use $p_v$ to denote the parent node of $v$ in the tree. We have a variable $w_t(v)$ for each vertex; which is 1 if at round $t$ either $v$ was newsly visited or it is in the span of the current boundary.

$$
\begin{array}{lllll}
 & \min\ D & & & \\
\text{s.t.} & & & & \\
 & x_t(r) & = & 1 & \text{for root } r \text{ and } \forall t \geq 0 \\
\text{(monotonicity)} & x_t(p_v) & \geq & x_t(v) & \forall t \geq 0, v \in V \\
\text{(progress)} & x_{t+1}(v) & \geq & x_t(v) & \forall t \geq 0, v \in V \\
 & w_t(v) & \geq & x_t(v) - x_{t-1}(u) & \forall t \geq 0, \forall u \in T_v \\
\text{(work)} & \sum_{v \in V} w_t(v) & \leq & D & \forall t \geq 0
\end{array}
$$

However, the LP has $\Omega(\sqrt{n})$ gap for the 3-spider example. It might be possible to strengthen the linear program.

# Chapter 5

# Weighted Bandwidth

In this section, we consider a slightly different notion of distortion. For the given metric space $(V, d)$, we ask for a mapping $f : V \to \{1, 2, \ldots, n\}$ instead of asking for a non-contracting embedding. Such a map $f$ is called a *linear ordering*. The stretch of the linear ordering $f$ is defined as

$$\max_{(x,y)} \frac{|f(x) - f(y)|}{d(x,y)}.$$

We also consider the following generalization of the problem.

**Weighted Bandwidth** Consider a graph $G = (V, E)$ on $n$-vertices with edge weights $w : E \to \mathbb{R}$. Let $f : V \to [1, n]$ be a 1-1 map. Such a map $f$ is called a *linear arrangement*. The weighted bandwidth of the linear arrangement $f$ is defined as the maximum stretch of any edge, i.e.

$$\mathsf{bw}(f) = \max_{(i,j)\in E} w(i,j) \cdot |f(i) - f(j)|.$$

The weighted bandwidth of the graph $G$ is the minimum possible bandwidth achievable by any linear arrangement $f : V \to [1, n]$.

$$\mathsf{bw}(G) = \min_{f:V\to[1,n]} \mathsf{bw}(f).$$

The goal is to find a linear arrangement of the vertices of $G$ which minimizes the weighted bandwidth.

The problem of minimizing bandwidth (i.e. when all the weights are 1) was shown to be NP-hard by Papadimitriou [Papadimitriou, 1976]. Blum et al. [Blum et al., 2000] gave an SDP relaxation of the bandwidth and other linear ordering problems. The first non-trivial approximation to this problem was given by Feige [Feige, 2000]. Subsequently, Dunagan and Vempala [Dunagan and Vempala, 2001] showed how to improve the approximation factor based on the SDP relaxation of Blum et al. Recently, Krauthgamer et al. [Krauthgamer et al., 2004] showed an algorithm to construct volume respecting embeddings and thus reduced the approximation factor to $O(\log^3 n)$.

The main result of this section is summarized in the following theorem.

**Theorem 15** *There exists a polynomial-time algorithm that produces $O(\log^2 n \log n\Delta)$-approximation the minimum weighted bandwidth problem.*

**Techniques** The main idea in our algorithm is similar to that of Feige [Feige, 2000]. However, instead of using shortest path metric on graph $G$, we use a different metric $(V, d)$ that is constructed using weights of the edges. The stretch of a linear ordering of $(V, d)$ is exactly same as the weighted bandwidth of graph $G$. We embed the metric $(V, d)$ into Euclidean space using a volume respecting embedding. Finally, we project the points on a random line. The ordering of the points on the random line is output as the ordering of the vertices.

In the metric $(V, d)$, the length of an edge $(u, v)$ is $1/w(u, v)$. We construct the shortest path metric on $V$ using these lengths. We also define a new lower bound for the weighted bandwidth. The construction of the metric $(V, d)$ lets us bound the weighted bandwidth of the output with respect to the lower bound.

**Definitions and Notation** We first define a few quantities that we shall use in the analysis later.

Throughout the discussion, let $N(\mu, \sigma^2)$ denote the Gaussian distribution with mean $\mu$ and standard deviation $\sigma$. We use the following simple fact about the Gaussian distribution.

**Fact 5.1** *Let $x \sim N(\mu, \sigma^2)$ be a random variable. Let $I$ be an interval of length $l$.*

$$\mathbf{Pr}[x \in I] \ \leq \ \frac{l}{\sigma}$$

We define *Tree Volume* $\mathsf{Tvol}()$ of a metric as the product of the edge lengths of the minimum spanning tree on the metric. Let aff$(x_1, x_2, \ldots, x_k)$ denote the affine span of the points $x_1, \ldots, x_k \in \mathbb{R}^m$. An $(\eta, k)$*–well-separated embedding* of a metric $(V, d)$ is a contracting map $\phi : V \to \mathbb{R}^n$ that satisfies the following condition.

For each set $S \subseteq V$, s.t. $|S| = k$, there exists a permutation $\{s_0, s_1, \ldots, s_{k-1}\}$ of $S$ such that, for all $i$, if $L_i = \text{aff}(\phi(s_0), \ldots, \phi(s_{i-1}))$, then

$$\text{dist}(\phi(s_i), L_i) \geq \frac{1}{\eta}\, d(s_i, \{s_0, \ldots, s_{i-1}\}) \tag{5.1}$$

The notion of well-separated embeddings is very closely related to the that of volume-respecting embeddings. We use the following result by Krauthgamer et al. [Krauthgamer et al., 2004].

**Lemma 28** *([Krauthgamer et al., 2004]) There exists an algorithm to construct* $(\log n, k)$*–well-separated embedding for every k:* $2 \leq k \leq n$.

## 5.1 Algorithm

Our algorithm is based on a metric on the graph $G$ derived from the weights. We construct this metric as follows. Given the weights $w$ on edges of the graph $G$, we define the metric $d$ on the graph as follows. Let $l(e) = \frac{1}{w(e)}$ denote the length of an edge $e$. The metric completion results in an instance where the weights $w(e)$ of some edges have *increased*. Using these lengths, we define the distance between any pair $u, v$ as $d(u, v) =$ the length of shortest path (according to lengths $l(\cdot)$) from $u$ to $v$.

## 5.2 Analysis

We first give a lower bound on the minimum weighted bandwidth based on the metric $(V, d)$. Throughout this section we assume that $d(u, v) \geq 1$ for all vertices $u, v \in V$. This can be easily achieved by scaling all the weights equally and hence it doesn't change the approximation factor.

**Lower Bound** Let $B(v, r)$ denote the ball of radius $r$ centered at vertex $v$ under metric $d$, i.e. $B(v, r) = \{u \in V \mid d(u, v) \leq r\}$. Let $|B(v, r)|$ denote the number of vertices in the set $B(v, r)$. Define the local density of the metric $(G, d)$ as

$$D \;=\; \max_{v,r} \frac{|B(v, r)|}{2r}.$$

**1:** For each edge $e$, define its length as $l(e) := \frac{1}{w(e)}$. For each pair of vertices $(u, v)$, let $d(u, v) =$ be the length of shortest path according the lengths $l(e)$. Scale the distances, so that the minimum pairwise distance is 1.

**2:** Let $\phi : V \to \mathbb{R}^n$ be an $(\eta, k)$–well-separated embedding of the metric $(V, d)$ into $\ell_2$.

**3:** Let $\vec{r} = (r_1, r_2, \ldots, r_n)$ be a vector in $\mathbb{R}^n$, where $r_i$ is a random variable with distribution $N(0, 1)$, for $i = 1, \ldots, n$. Let $\pi : \mathbb{R}^n \to \mathbb{R}$ be a map defined by

$$\pi(\vec{v}) \ = \ \vec{r} \cdot \vec{v}$$

**4:** Let $\psi : V \to \mathbb{R}$ be a map defined by

$$\psi(v) \ := \ \pi\left(\phi(v)\right) \qquad \text{for each } v \in V$$

**5:** Output a linear arrangement according to the ordering obtained by the map $\psi$.

Figure 5.1: Algorithm Weighted-Bandwidth (WB)

**Claim 5.2** *$D$ is a lower bound on* $\mathsf{bw}(G)$.

**Proof.** Fix a vertex $v$ and a radius $r$ and consider the ball $B(v, r)$. In any linear arrangement, the number of vertices between the leftmost vertex $u$ and the rightmost vertex $w$ is at least $|B(v, r)|$. Since, both $u$ and $w$ belong to $B(v, r)$, the distance between them is at most $2r$. Consider a shortest path (under lengths $l(e)$) joining $u$ and $w$ in graph $G$. For all edges $e'$ of this shortest path, we have $d(e') = l(e') = \frac{1}{w(e')}$.

At least one of the edges in this path has stretch $\geq \frac{|B(v,r)|}{2r}$. Here, stretch of an edge $e'$ is

$$\frac{\#(\text{vertices between endpoints of } e')}{l(e')}.$$

Hence the weighted bandwidth of graph $G$ is at least $D$. ■

The analysis of the algorithm is very similar to that of Feige's algorithm [Feige, 2000]. We bound the stretch of any edge in the random projection instead of its length.

**Lemma 29** *Consider a set $S \subseteq V$ with $|S| = k$. The probability that $S$ gets mapped* *inside an interval of length $\lambda$ under the map $\psi$ is bounded as follows.*

$$\mathbf{Pr}[S \textit{ gets mapped in an interval}] \leq \frac{O(\eta\lambda)^{k-1}}{\mathsf{Tvol}(S)}$$

**Proof.**

Fix a set $S = \{s_0, s_1, \ldots, s_{k-1}\}$. Let's call $\phi(s_i)$ as $v_i$. Let $L_i = \text{span}\{v_0, \ldots, v_{i-1}\}$.

We use the spherical symmetry of choosing a random line. Using suitable rotation and translation, we assume that $v_0 = \vec{0}$ and $L_i = \text{span}\{\vec{e_1}, \ldots, \vec{e_i}\}$.

We can now interpret the well-separatedness property of the map $\phi$. Let $v_i = (v_{i1}, \ldots, v_{ii}, 0, \ldots, 0)$ and let $d_G(s_i, \{s_0, \ldots, s_{i-1}\}) = q_i$. Then well-separatedness says that

$$v_{ii} \geq \frac{q_i}{\eta}$$

Note that the map $\psi$ can be described as follows.

$$\psi(x) = \langle \phi(x), \vec{r} \rangle,$$

where $\vec{r} = (r_1, \ldots, r_d)$ and each $r_i \sim N(0, 1)$. Thus we have, $\psi(s_i) = \langle \vec{v_i}, \vec{r} \rangle$ for $i = 0, 1, \ldots, k-1$.

We now bound the probability that all of $S$ is mapped into the interval $I = [0, l)$. We write this probability is the product of conditional probabilities.

$$\mathbf{Pr}[\psi(S) \subseteq I] = \prod_{i=0}^{k-1} \mathbf{Pr}[\psi(s_i) \in I \mid \psi(\{s_0, \ldots, s_{i-1}\}) \subseteq I] \qquad (5.2)$$

Now we bound $\mathbf{Pr}[\psi(s_i) \in I \mid \psi(\{s_0, \ldots, s_{i-1}\}) \subseteq I]$. Note that $\psi(s_i) = \sum_{j=1}^{i} v_{ij} r_j$. In order that $\psi(s_i) \in I$, we need to have $v_{ii} r_i \in I'$, for some other interval $I'$ of length $l$. All of the $\psi(s_0), \ldots, \psi(s_{i-1})$ are independent of $r_i$. Thus

$$\begin{aligned}
\mathbf{Pr}[\psi(s_i) \in I \mid \psi(s_0) \in I, \ldots, \psi(s_{i-1}) \in I] &\leq \mathbf{Pr}[v_{ii} r_i \in I'] \\
&\leq \frac{\lambda}{v_{ii}} \\
&\leq \frac{\eta\lambda}{q_i}
\end{aligned}$$

Here, the second inequality follows from the Fact 5.1 and the final one from well-separatedness of map $\phi$.

Now we can simplify equation (5.2).

$$\begin{aligned}\mathbf{Pr}[\psi(S) \subseteq I] &\le \prod_{i=1}^{k-1} \frac{\eta l}{q_i} \\ &\le \frac{(\eta l)^{k-1}}{\mathsf{Tvol}(S)}\end{aligned}$$

This proves the lemma. ■

**Lemma 30** *Stretch of any edge under the map $\psi$ is bounded by $\lambda$ with high probability.*

$$\mathbf{Pr}[\textit{Edge } e \textit{ has stretch} \ge \lambda] \le \frac{1}{2n^2}$$

**Proof.** Consider an edge $e = (u, w)$. Since the map $\phi$ is a contracting map, it does not stretch the edge. Therefore, to bound the stretch of the edge under map $\psi$, we need to bound its stretch under the projection $\pi$. Since $\pi$ is a linear map, we only need to consider stretch of a vector $\vec{v} \in \mathbb{R}^n$ due to the map $\pi$.

Recall that each coordinate of $\vec{r}$ follows the distribution $N(0, 1)$.

Because of spherical symmetry, it follows that $|\vec{r} \cdot \vec{v}|$ behaves like $N(0, \|\vec{v}\|^2)$. Let $X \sim N(0, 1)$ be a random variable.

$$\begin{aligned}\mathbf{Pr}\left[|\vec{r} \cdot \vec{v}| > 2\sqrt{\log n}\right] &= \mathbf{Pr}\left[\|v\| \cdot |X| > 2\sqrt{\log n}\right] \\ &\le \mathbf{Pr}\left[|X| > 2\sqrt{\log n}\right] \\ &\le \frac{2}{\sqrt{\log n} \times \sqrt{2\pi}} e^{-\frac{1}{2}(2\sqrt{\log n})^2} \\ &\le \frac{1}{2n^2}\end{aligned}$$

This concludes the proof. ■

We use the following lemma due to Feige.

**Lemma 31** *([Feige, 2000]) For all $S \subseteq V$, s.t. $|S| = k$, we have the following inequality.*

$$\frac{1}{\mathsf{Tvol}(S)} \le \sum_{\pi:[k]\to[k]} \frac{1}{d(v_{\pi(1)}, v_{\pi(2)}) \cdot \ldots \cdot d(v_{\pi(k-1)}, v_{\pi(k)})}$$

**Lemma 32** *For any metric $(V, d)$ we have the following.*

$$\sum_{S \subseteq V, |S|=k} \frac{1}{\mathsf{Tvol}(S)} \leq n \cdot O(D \log n\Delta)^{k-1},$$

*where $\Delta$ is the diameter of the metric.*

**Proof.** Using Lemma 31, we can write the following inequality.

$$\begin{aligned} \sum_{S:|S|=k} \frac{1}{\mathsf{Tvol}(S)} &\leq \sum_{S:|S|=k} \sum_{\pi:[k]\to[k]} \frac{1}{d(v_{\pi(1)}, v_{\pi(2)}) \cdot \ldots \cdot d(v_{\pi(k-1)}, v_{\pi(k)})} \\ &\leq \sum_{(u_1,u_2,\ldots,u_k)\in V^k} \frac{1}{d(u_1, u_2) \cdot \ldots \cdot d(u_{k-1}, u_k)} \end{aligned}$$

Let $\Delta$ be the diameter of the metric defined in Step 1 of the algorithm. Note that $\Delta \leq nW$, where $W$ is the ratio of maximum weight to minimum weight. To bound the sum on the right hand side, fix a vertex $u_1 \in V$ and a tuple $(a_1, a_2, \ldots, a_{k-1})$, with each $a_i \in \{1, 2, \ldots, \log \Delta\}$. Now, consider all the sequences of $k$ vertices $(u_1, u_2, \ldots, u_k)$, where the first vertex is $u_1$ and $d(u_i, u_{i+1}) \in [2^{a_i}, 2^{a_i+1})$ for all $i$.

We claim that the sum over this subset of sequences is $O(D)^{k-1}$. Note that any ball around a vertex $v$ of radius $r$ has at most $D \cdot (2r)$ vertices in it. Therefore, once we choose the vertex $u_i$ in the sequence, there are at most $O(D \cdot 2^{a_i})$ choices for $u_{i+1}$. Hence the total number of sequences in the above subset is at most $O(D)^{k-1} \cdot 2^{(\sum_i a_i)}$. The contribution of any such sequence to the sum is at most $1/2^{(\sum_i a_i)}$. Therefore, the sum over these subsets is at most $O(D)^{k-1}$.

Finally, there are $n$ choices for $u_1$ and $(\log \Delta)^{k-1}$ choices for the vector of $a_i$'s. Hence we get that the sum on RHS is at most $n \times O(D \log \Delta)^{k-1}$. Since $\Delta = O(nW)$, this concludes the proof.

■

Now we are ready to prove the main theorem.

**Proof of Theorem 1**: Using Lemmata 31 and 32, we get that $\#(\text{bad sets}) \leq n \cdot O(\eta\lambda\, D \log n)^{k-1}$. Furthermore, using Lemma 30, we assume that all edges have stretch $\leq \lambda$.

Let $B$ be the weighted bandwidth of the output of our algorithm. In particular, there is an edge $e$ whose weighted bandwidth is $B$. Therefore, the endpoints of the edge $e$ have $B/w(e)$ vertices between them in the linear ordering. Since, $l(e) = 1/w(e)$, there must be

$B \cdot l(e)$ vertices between the two endpoints of $e$ in the optimal linear ordering. However, in the random projection, stretch of the edge $e$ was bounded by $\lambda$, i.e. its length was at most $\lambda \cdot l(e)$. With this length, edge $e$ can span only $l(e)$ intervals of length $\lambda$ each. Thus, there is an interval of length $\lambda$ which has at least $\frac{B \cdot l(e)}{l(e)} = B$ points. Hence we can conclude that the number of bad sets of size $k$ is at least $\binom{B}{k}$.

Therefore, we get

$$\binom{B}{k} \leq n \cdot O(\eta\lambda\, D \log n\Delta)^{k-1}$$

Choosing $k = \log n$ and plugging in $\eta = \log n$ and $\lambda = \log n$ gives us the bound

$$B \leq O(\log^2 n \log n\Delta) \cdot D.$$

This proves the main theorem. $\square$

## 5.3 Discussion

It should be possible to replace the $\log n\Delta$ factor by a $O(\log n)$ factor. However, currently we do not know how to do it. A more difficult open question is to improve the approximation factor for the bandwidth problem.

# Chapter 6

# Spanning Tree Embeddings

In this chapter, we consider probabilistic embeddings into spanning tree metrics. The input metric is the shortest path metric on the input graph $G = (V, E)$. A probabilistic embedding into spanning trees is a probability distribution $\mathcal{T}$ over the spanning trees of the graph $G$. The quality of a probabilistic embedding is given by the *expected* distortion. It is defined as follows:

$$E[distortion] = \max_{u,v \in V} E_{T \in \mathcal{T}} d_T(u, v)/d(u, v)$$

Our goal is to find a probabilistic embedding with small expected distortion. Note that we are looking for uniform bounds on the expected distortion.

We give a simple randomized procedure that takes the shortest-path metric $d$ of a graph $G = (V, E)$, and whose output is a probabilistic embedding of $d$ into spanning subtrees of $G$. Our result is based on the techniques from Bartal [Bartal, 1996] combined with the recent results by Elkin et al. [Elkin et al., 2005].

For simplicity of exposition, we first consider the case when $G$ is an unweighted graph, and hence the diameter of $G$ is at most $(n - 1)$. The arguments can then be extended to the case of arbitrary edge-lengths using standard ideas (e.g. see [Bartal, 1996]).

While the guarantees of our algorithm are only marginally better than those of Elkin et al., we would like to point out that our improvements come from use of a different technique. It might be possible to combine the two approaches to improve the result further.

## 6.1 The Algorithm

Our randomized algorithm uses the idea of star decompositions proposed by Elkin et al..

**Definition 1 (star-decomposition)** *A star-decomposition of a graph $G$ with a designated root node $r_0$ is a set of disjoint connect components of $G_0 = (V_0, E_0), \ldots, G_k = (V_k, E_k)$ together with a collection of root nodes $r_0, \ldots, r_k$ such that $r_i \in V_i$ for all $0 \leq i \leq k$ and each $r_i$, $1 \leq i \leq k$ has a neighbor in $V_0$.*

The procedure of Figure 6.1 takes a graph $G$ with a root $r$, and outputs a star-decomposition of $G$. For the backward cut step in the algorithm, we define a new distance function: *backward-edge* distance. For the definition of the new distance function we replace each edge in $G \setminus V_0$ by two directed edges in opposite direction. We define the length $\ell(u, v)$ of such a directed edge $(u, v)$ as

$$\ell(u,v) = \begin{cases} 1 & \text{if } d(r_0, v) = d(r_0, u) - 1 \\ 1 & \text{if } d(r_0, v) = d(r_0, u) \\ 0 & \text{if } d(r_0, v) = d(r_0, u) + 1 \end{cases}$$

Using this length function, we get the shortest path distance that we call the *backward-edge distance*. The distance from $x$ to $y$ counts how often an edge has to be used in backward or sidewards direction according to distance from $r_0$ in order to reach $y$ from $x$. (Note that we only used directed edges to define the new distance function on $X$. In the following all edges are undirected again.)

Given the above procedure to find random star-decompotions of $G$, the embedding of $G$ into random spanning trees is the same as in Elkin et al. [Elkin et al., 2005].

The following is the main theorem of the paper:

**Theorem 16** *The algorithm* Embed-Tree *induces a distribution over spanning trees of $G$ such that the expected stretch of each edge of $G$ is $O(\log^2 n)$.*

Let us give a roadmap for the proof. We first prove two simple lemmas that bound the probability that an edge $(u, v)$ is cut by the forward and backward cut steps of Random-Star-Decomp; each of these probabilities will turn out to be $O(\log n/\Delta)$. Furthermore, it is easy to see that the number of levels encountered by any edge is $O(\log n)$. In previous such analyses (e.g., by Bartal [Bartal, 1996]), the argument used is that if $(u, v)$ is cut at level $\Delta$, the distance between $u$ and $v$ in the final tree is at most $O(\Delta)$, and hence the total stretch incurred by the edge is $O(\log^2 n)$. In our case, it is true that the distance between $u$

1: **Input:** Graph $G$ with root $r$ such that $\mathrm{radius}_r(G) = \Delta$.

2: Pick a distance $\gamma$ uniformly at random from the interval $[\Delta/4, \Delta/2]$.

3: **(Forward Cut)** Cut all the edges at distance $\gamma$ from the root $r$; let $V_0$ be the component containing the root $r$.

4: Let $x_1, x_2, \ldots, x_k$ be the "portal" vertices in $G \setminus V_0$ whose neighboring edges have been cut in Step 2, and let $p(x_i)$ be some "parent" of $x_i$ in $V_0$.

5: Let $X \leftarrow V \setminus V_0$ be the "remaining" vertices.

6: **(Backward Cut)** For $i = 1, 2, \ldots$, consider the portal vertex $x_i \in X$, and choose a random distance $R$ from the distribution $\Delta \cdot Exp\{6 \log n\}$. Cut all the edges lying at *backward-edge distance* $R$ from vertex $x_i$ to get the component $V_i$.

7: Let $X \leftarrow X \setminus V_i$; if $X \neq \emptyset$, goto Step 6.

8: Add the edges $(x_i, p(x_i))$ to get the graph $G_1$.

Figure 6.1: Algorithm Random-Star-Decomp $(G, r)$

1: **Input:** Graph $G$ with root $r$ such that $\mathrm{radius}_r(G) = \Delta$.

2: Execute Random-Star-Decomp $(G, r)$ to get the components $V_0, V_1, \ldots, V_k$, with each $V_i$ being rooted at $r_i$, and each $V_i$ (for $i \geq 1$) connected to $V_0$ by edges $(r_i, p(r_i))$.

3: Execute Embed-Tree $(V_i, r_i)$ recursively to convert each $V_i$ into a subtree of $G[V_i]$.

Figure 6.2: Algorithm Embed-Tree $(G, r)$

and $v$ when they are separated is indeed $O(\Delta)$, but it may increase over the course of later cuts; hence we need Theorem 17 to say that the $u$-$v$ distance in the tree does remain $O(\Delta)$ with high probability.

### 6.1.1 Edge-Cutting Probabilities and Recursion Depth

**Claim 6.1** *The probability that an edge is cut by a forward cut at level $\Delta$ is at most $O(\frac{1}{\Delta})$.*

**Proof.** Consider an edge $e = (u, v)$. If $v$ is closer than $\Delta/4$ or farther than $\Delta/2$ from the root $r$, then $(u, v)$ cannot be cut by the forward cut step. Otherwise $e$ will be cut iff $\gamma \in [d(r, u), d(r, v)]$. Since $\gamma \in_{\text{u.a.r}} [\Delta/4, \Delta/2]$, the probability of this event is at most $4/\Delta$. ■

**Claim 6.2** *The probability that an edge is cut by a backward cut is at most $\frac{6 \log n}{\Delta}$.*

**Proof.** Recall that in Backward Cut step, we choose a random radius using the distribution $\Delta \cdot Exp\{6 \log n\}$. Consider the ball $V_1$ centered at the vertex $r_1$ that has the radius $X_1 \sim \Delta \cdot Exp\{6 \log n\}$. This ball cuts an edge $e = (u, v)$, if exactly one of $u$ and $v$ is inside. Without loss of generality, assume that $u$ is closer to $r_1$ than $v$. We consider three separate cases. If $X_1 \geq d(r_1, v)$, then the edge $(u, v)$ is inside the ball $V_1$ and won't be cut. If $d(r_1, u) \leq X_1 < d(r_1, v)$, then the ball $V_1$ cuts the edge $e$. Finally, if $X_1 < d(r_1, u)$, then the edge $e$ is outside ball $V_1$ and some other ball can cut it. Let $p$ denote the probability that the edge $e$ is cut. Then we have

$$p \leq \mathbf{Pr}[d(r_1, u) \leq X_1 < d(r_1, v)] + \mathbf{Pr}[X_1 < d(r_1, u)] \cdot p$$

Since the edges are unweighted: $d(r_1, v) \leq d(r_1, u) + 1$. Let $d = d(r_1, u)$. Then we have

$$p \ \leq \ \frac{\mathbf{Pr}[d \leq X_1 \leq d + 1]}{1 - \mathbf{Pr}[X_1 < d]} \ \leq \ \mathbf{Pr}[X_1 \leq d + 1 \mid X_1 \geq d] \ \leq \ \frac{6 \log n}{\Delta}$$

The last step follows from the memoryless property of the exponential distribution. ■

Since an edge is cut either as a forward edge or as a backward edge, we get the following result.

**Corollary 6.3** *The probability that an edge $(u, v)$ is cut at level $\Delta$ is at most $\frac{7 \log n}{\Delta}$.*

**Fact 6.4** *The depth of the recursion for* Embed-Tree *is $O(\log n)$* **whp**.

**Proof.** To prove this, we claim that **whp** the diameter of each of the $V_i's$ is at most $11\Delta/12$. Indeed, since we choose the expected value of the exponential distribution to be suitably small, the diameter reduces by a constant factor whp. Hence the recursion depth will be $O(\log_{12/11} \operatorname{diam}(G)) = O(\log n)$. ■

### 6.1.2 Bounding Additional Stretch

**Theorem 17** *Given any $G$ rooted at $r$, and any vertex $v \in V(G)$, the distance from $v$ to $r$ in the random tree produced by* Embed-Tree *$(G, v)$ is $O(d_G(v, r))$* **whp**.

**Proof.** We can view the expansion of diameter of $G$ as follows. In each level of recursion, we run the procedure Random-Star-Decomp in each of the components. Consider the graph after $i$ levels of recursive application of Random-Star-Decomp. Let the diameter after level $i$ be denoted by $\Delta_i$. Let there be $m$ components on the root to leaf path at this stage, with diameters $D_1, D_2, \ldots, D_m$.

Note that one application of Random-Star-Decomp increases the diameter of the $j^{th}$ component from $D_j$ to $D_j + D_j X_j$, where $X_j \sim Exp\{O(\log n)\}$. Hence, we can bound the diameter $\Delta_{i+1}$ of the graph after $i^{\text{th}}$ recursive application of Random-Star-Decomp as

$$\Delta_{i+1} \leq \Delta_i (1 + \sum_j \alpha_j X_j)$$

where $\alpha_j = D_j / \Delta_i$. Note that $\alpha_j$ depends on the history of the algorithm so far. Nevertheless, as we show later in Lemma 33, we can get the following:

$$\Delta_{i+1} \preceq \Delta_i \cdot (1 + 2(Y_i + 1/\log n))$$

where $Y_i \sim Exp\{O(\log n)\}$ and $Y_i$ is independent of the history.

Thus the final diameter (after $k$ steps) of the graph is bounded as follows:

$$\begin{aligned}
\Delta_k &\preceq \Delta \cdot \prod_i (1 + 2(Y_i + 1/\log n)) && (6.1)\\
&\leq \Delta \cdot \exp\{\sum_{i=1}^{k} 2(Y_i + 1/\log n))\} && (6.2)\\
&\leq \Delta \cdot \exp\{O(1)\} \qquad \text{w.h.p. (From Lemma 34)} && (6.3)
\end{aligned}$$

■

**Proof of Theorem 16**: Let us first consider all the bad events (that the diameter of some $V_i$ is too large, and that the distance between $u$ and $v$ gets too large. Since there are only (at most) $n^3$ such bad events, each one happens with probability at most $1/\operatorname{poly}(n)$, and the distortion suffered when any of these bad events happen is at most $n$, we can ignore all these events.

From Theorem 17 it follows that an edge $e$ that gets cut at level $\Delta$ suffers a distortion of $O(\Delta)$. Therefore we can compute the expected distortion of edge $e$ as follows.

$$\begin{aligned}
Distortion(e) &= \sum_{\text{level } \Delta} \mathbf{Pr}[\text{Edge } e \text{ cut at level } \Delta] \cdot O(\Delta)\\
&\leq \sum_{\text{level } \Delta} O(\frac{\log n}{\Delta}) \cdot O(\Delta)\\
&= \sum_{\text{level } \Delta} O(\log n)\\
&= O(\log^2 n)
\end{aligned}$$

□

## 6.2 Stochastic Domination and Tail Bounds

In this section, we prove two simple yet crucial lemmas: the first states that a convex combination of exponential i.i.d. random variables is stochastically dominated by (a suitably shifted version) of one independent copy of the random variable.

**Definition 2** *A random variable $X$ is stochastically dominated by another random variable $Y$, if the following holds for all $t$.*

$$\mathbf{Pr}[X \geq t] \leq \mathbf{Pr}[Y \geq t].$$

**Proposition 6.5** *The following facts hold:*

•*If $X \preceq Y$ and $Y \preceq Z$, then $X \preceq Z$.*

•*Let $X, Y$, and $Z$ be independent r.v.s, with $Z \geq 0$. If $X \preceq Y$, then $XZ \preceq YZ$.*

**Proof.** Recall that $X \preceq Y$ holds iff $Pr[X \geq t] \leq Pr[Y \geq t]$ for all $t$. The first property follows from the fact that

$$\mathbf{Pr}[X \geq t] \leq \mathbf{Pr}[Y \geq t] \leq \mathbf{Pr}[Z \geq t]$$

To prove the second property, assume that $f(z)$ is the probability density function for the random variable $Z$. Then we have

$$\mathbf{Pr}[XZ \geq t] = \int \mathbf{Pr}[Xz \geq t] f(z) dz \leq \int \mathbf{Pr}[Yz \geq t] f(z) dz = \mathbf{Pr}[YZ \geq t].$$

■

In the next lemma, we bound a convex combination of exponentially distributed random variables.

**Lemma 33** *Let $X_0, X_1, \ldots, X_m \sim Exp\{\lambda\}$ be i.i.d. random variables. And let $\alpha_1, \alpha_2, \ldots, \alpha_m$ be $m$ non-negative real numbers such that, $\sum_i \alpha_i = 1$. Then we have: $\sum_i \alpha_i X_i \preceq 2(X_0 + 1/\lambda)$, or equivalently*

$$Pr[\sum_{i=1}^{m} \alpha_i X_i \geq t + 2/\lambda] \leq Pr[2X_0 \geq t] \qquad \textit{for all } t \geq 0$$

**Proof.** To prove an upper bound on LHS, we use the moment generating function $M(s)$ of $\sum_i \alpha_i X_i$. Recall that the moment generating function of a random variable $X$ is defined as $\exp e^{sX}$. Using this definition, we get

$$M(s) = \prod_i \frac{1}{1 - \alpha_i (s/\lambda)}$$

We first bound the maximum possible value of $M(s)$ subject to the constraint that $\sum_i \alpha_i = 1$.

**Fact 6.6** *The following is true for all $\beta, \gamma \geq 0$ such that $\beta + \gamma \leq 1$ and* $0 < t < 1$.

$$\frac{1}{(1-\beta t)(1-\gamma t)} \leq \frac{1}{1-(\beta+\gamma)t}.$$

Using this simple fact repeatedly for $M(s)$, we get $M(s) \leq \frac{1}{1-(s/\lambda)}$. Now using the Markov inequality,

$$Pr[\sum_{i=1}^{m} \alpha_i X_i \geq t + 2/\lambda] \leq e^{-s(t+2/\lambda)} M(s) \qquad \text{for all } s \geq 0$$

Finally, choosing $s = \lambda/2$, we get

$$LHS \leq \frac{2}{e} e^{-(\lambda t/2)} \ \leq \ Pr[2X_0 \geq t],$$

which proves the lemma. ∎

The second is a standard tail bound on the sum of i.i.d. exponential random varaibles.

**Lemma 34** *Let $X_1, X_2, \ldots, X_k \sim Exp\{\log n\}$ be i.i.d. random variables. Let $k \leq \log n$. Then*

$$Pr[\sum_i X_i \geq 4] \leq \frac{1}{n^2}$$

**Proof.** Without loss of generality assume that $k = \log n$. Let $M(s)$ denote the moment generating function of $\sum_i X_i$. Using standard techniques, we get

$$M(s) = \prod_i \frac{1}{1-(s/k)} = (\frac{1}{1-(s/k)})^k$$

Using Chernoff's inequality,

$$Pr[\sum_i X_i \geq 4] \leq e^{-4s} (\frac{1}{1-(s/k)})^k \qquad \text{for all } 0 < s < k.$$

Now choosing $s = 3k/4$ (so as to optimize the upper bound), we get

$$Pr[\sum_i X_i \geq 4] \leq \left(\frac{4}{e^3}\right)^k \leq \frac{1}{n^2}$$

∎

## 6.3 Discussion

The big open question is whether it is possible get $O(\log n)$ expected distortion. It will also be interesting to give approximation algorithm for the expected distortion.

# Chapter 7

# Conclusion

In this dissertation, we initiated the study of metric embeddings from an approximation algorithm point-of-view. We focused on the host metric being the real line.

For embedding an arbitrary into the line metric, the existing results give a uniform bound of $O(n)$ on the distortion. To cope with this bound, our goal was to give approximation algorithm for the distortion.

However there are many open problems in this framework. Finding the best embedding of an arbitrary metric into $\ell_p$ metric is one; Finding the best probabilistic embedding of an arbitrary metric into (spanning) tree metrics is another.

Another way to cope with high uniform bound is to consider metric embeddings with $\epsilon$-slack Kleinberg et al. [2004], Chan et al. [2005], where distortion is small for all but an $\epsilon$ fraction of the pairwise distances. As an example, consider the uniform metric on $n$ points (i.e. all pairwise distances are 1). To embed this metric into the line metric, we need $\Omega(n)$ distortion. However, there is a simple embedding of this metric into the line with $1/\epsilon$ distortion and $\epsilon$-slack.

While we studied embedding into line metrics (one dimensional Euclidean space), more progress needs to be done on embedding into the Euclidean space with a small, fixed number of dimensions. It will be useful as an alternative to heuristics for multi-dimensional scaling in dimension reduction of data. These results could perhaps be combined with the $\epsilon$-slack results to be more useful in practice.

# Bibliography


Amit Agarwal, Moses Charikar, Konstantin Makarychev, and Yury Makarychev. $o(\sqrt{\log n})$ approximation algorithms for min uncut, min 2cnf deletion, and directed cut problems. In *Proceedings of the 37th ACM Symposium on the Theory of Computing (STOC)*, pages 573–581, 2005. 3, 3.5

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 (electronic), 1999. ISSN 1095-7111. 1, 1.2, 3.2.1

Noga Alon, Richard M. Karp, David Peleg, and Douglas West. A graph-theoretic game and its application to the $k$-server problem. *SIAM J. Comput.*, 24(1):78–100, 1995. ISSN 0097-5397. 1.5

Aaron Archer and David P. Williamson. Faster approximation algorithms for the minimum latency problem. In *Proceedings of the 14th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 88–96, 2003. 1.1

S. Arora. Polynomial time approximation schemes for Euclidean traveling salesman and other geometric problems. *J. ACM*, 45(5):753–782, 1998. 1

Sanjeev Arora. Polynomial time approximation schemes for Euclidean TSP and other geometric problems. In *37th Annual Symposium on Foundations of Computer Science (Burlington, VT, 1996)*, pages 2–11. IEEE Comput. Soc. Press, Los Alamitos, CA, 1996. 1

Sanjeev Arora and George Karakostas. Approximation schemes for minimum latency problems. *SIAM J. Comput.*, 32(5):1317–1337 (electronic), 2003. ISSN 1095-7111. 3, 1.1, 2.2, 2.2

Yonatan Aumann and Yuval Rabani. An $O(\log k)$ approximate min-cut max-flow theorem and approximation algorithm. *SIAM J. Comput.*, 27(1):291–301, 1998. ISSN 1095-7111. 1

Mihai Bădoiu, Julia Chuzhoy, Piotr Indyk, and Anastasios Sidiropoulos. Low-distortion embeddings of general metrics into the line. In *Proceedings of the 37th ACM Symposium on the Theory of Computing (STOC)*, 2005a. 1.3

Mihai Bădoiu, Kedar Dhamdhere, Anupam Gupta, Yuri Rabinovich, Harald Räcke, R. Ravi, and Anastasios Sidiropoulos. Approximation algorithms for embeddings into low-dimensional spaces. In *Proceedings of the 16th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 119–128, 2005b. 1.3

Mihai Bădoiu, Piotr Indyk, and Yuri Rabinovich. Approximate algorithms for embedding metrics into low-dimensional spaces. In *Unpublished manuscript*, 2003. 1.2

Yair Bartal. Probabilistic approximations of metric spaces and its algorithmic applications. In *Proceedings of the 37th Symposium on the Foundations of Computer Science (FOCS)*, pages 184–193, 1996. 1, 1.5, 1.5, 6, 6.1

Yair Bartal. On approximating arbitrary metrics by tree metrics. In *Proceedings of the 30th ACM Symposium on the Theory of Computing (STOC)*, pages 161–168, 1998. 1.5

Yair Bartal, Avrim Blum, Carl Burch, and Andrew Tomkins. A polylog($n$)-competitive algorithm for metrical task systems. In *Proceedings of the 29th ACM Symposium on the Theory of Computing (STOC)*, pages 711–719, 1997. 1

Avrim Blum, Prasad Chalasani, Don Coppersmith, Bill Pulleyblank, Prabhakar Raghavan, and Madhu Sudan. The minimum latency problem. In *Proceedings of the twenty-sixth annual ACM symposium on Theory of computing*, pages 163–171. ACM Press, 1994. ISBN 0-89791-663-8. 1.1, 3, 1.1, 2.2

Avrim Blum, Goran Konjevod, R. Ravi, and Santosh Vempala. Semi-definite relaxations for minimum bandwidth and other vertex-ordering problems. *Theoret. Comput. Sci.*, 235(1):25–42, 2000. ISSN 0304-3975. (Preliminary version in *30th STOC*, pages 100–105, 1998). 1, 1.4, 5

Jean Bourgain. On Lipschitz embeddings of finite metric spaces in Hilbert space. *Israel Journal of Mathematics*, 52(1-2):46–52, 1985. 1

F. Buckley and F. Harary. *Distance in Graphs*. Addison Wesley Reading, MA, 1990. 2.3

Gruia Călinescu, Howard Karloff, and Yuval Rabani. Approximation algorithms for the 0-extension problem. In *Proceedings of the twelfth annual ACM-SIAM symposium on Discrete algorithms*, pages 8–16. ACM Press, 2001. ISBN 0-89871-490-7. 1

L. Cavalli-Sforza and A. Edwards. Phylogenetic analysis models and estimation procedures. *American Journal of Human Genetics*, 19:233–257, 1967. 1.2

Hubert T-H. Chan, Kedar Dhamdhere, Anupam Gupta, Jon M. Kleinberg, and Alexandrs Slivkins. On metric embeddings with slack. in submission, 2005. 7

Kamalika Chaudhuri, Brighten Godfrey, Satish Rao, and Kunal Talwar. Paths, trees and minimizing latency. In *Proceedings of the 44th Symposium on the Foundations of Computer Science (FOCS)*, pages 36–45, 2003. 1.1, 2.1.2

Michel Marie Deza and Monique Laurent. *Geometry of cuts and metrics*, volume 15 of *Algorithms and Combinatorics*. Springer-Verlag, Berlin, 1997. ISBN 3-540-61611-X. 1.2

Kedar Dhamdhere. Approximating additive distortion of embeddings into line metrics. In *Proceedings of the International Workshop on Approximation Algorithms for Combinatorial Optimization Problems (APPROX)*, 2004. 1.2

Kedar Dhamdhere. A note on minimizing weighted bandwidth of a graph. Submitted, 2005. 1.4

Kedar Dhamdhere, Anupam Gupta, and Harald Räcke. Improved embeddings into spanning trees. in submission, 2006. 1.5

Kedar Dhamdhere, Anupam Gupta, and R. Ravi. Approximation algorithms for minimizing average distortion. In *21st Annual Symposium on Theoretical Aspects of Computer Science (STACS)*, volume 2996 of *Lecture Notes in Computer Science*, pages 234–245, 2004. ISBN 3-540-21236-1. 1.1

John Dunagan and Santosh Vempala. On Euclidean embeddings and bandwidth minimization. In *Approximation, randomization, and combinatorial optimization (Berkeley, CA, 2001)*, volume 2129 of *Lecture Notes in Comput. Sci.*, pages 229–240. Springer, Berlin, 2001. 1.4, 5

Michael Elkin, Yuval Emek, Dan Spielman, and Shang-Hua Teng. Lower-stretch spanning trees. In *Proceedings of the thirty-seventh ACM symposium on Theory of computing (STOC)*, 2005. (document), 1.5, 1.5, 6, 6.1

Yuval Emek and David Peleg. Approximating minimum max-stretch spanning trees on unweighted graphs. In *SODA '04: Proceedings of the fifteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 261–270, Philadelphia, PA, USA, 2004. Society for Industrial and Applied Mathematics. ISBN 0-89871-XXX-X. 1.5

Jittat Fakcharoenphol, Chris Harrelson, and Satish Rao. The k-traveling repairman problem. In *Proceedings of the fourteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 655–664. Society for Industrial and Applied Mathematics, 2003a. ISBN 0-89871-538-5. 1.1, 2, 1.1, 2.1.2, 2.1.2

Jittat Fakcharoenphol, Satish Rao, and Kunal Talwar. A tight bound on approximating arbitrary metrics by tree metrics. In *Proceedings of the thirty-fifth ACM symposium on Theory of computing*, pages 448–455. ACM Press, 2003b. ISBN 1-58113-674-9. 1, 1.5

M. Farach, S. Kannan, and T. Warnow. A robust model for finding optimal evolutionary trees. *Algorithmica*, 13(1-2):155–179, 1995. ISSN 0178-4617. 1, 1.2

Uriel Feige. Approximating the bandwidth via volume respecting embeddings. *J. Comput. System Sci.*, 60(3):510–539, 2000. ISSN 0022-0000. 30th Annual ACM Symposium on Theory of Computing (Dallas, TX, 1998). 1, 1.4, 5, 5, 5.2, 31

Michael R. Garey and David S. Johnson. *Computers and Intractability: A guide to the theory of NP-completeness*. W. H. Freeman and Company, San Francisco, 1979. 3.6, 4.3

Naveen Garg, Goran Konjevod, and R. Ravi. A polylogarithmic approximation algorithm for the group Steiner tree problem. *Journal of Algorithms*, 37(1):66–84, 2000. (Preliminary version in *9th SODA*, pages 253–259, 1998). 1

Naveen Garg, Vijay V. Vazirani, and Mihalis Yannakakis. Approximate max-flow min-(multi)cut theorems and their applications. *SIAM J. Comput.*, 25(2):235–251, 1996. ISSN 0097-5397. 3.4.1, 2, 3.4.1

Michel Goemans and Jon Kleinberg. An improved approximation ratio for the minimum latency problem. *Math. Programming*, 82(1-2, Ser. B):111–124, 1998. ISSN 0025-5610. Networks and matroids; Sequencing and scheduling. 1.1, 1.1

Anupam Gupta. Steiner points in tree metrics don't (really) help. In *Proceedings of the Twelfth Annual ACM-SIAM Symposium on Discrete Algorithms (Washington, DC, 2001)*, pages 220–227, Philadelphia, PA, 2001. SIAM. 1

Johan Håstad, Lars Ivansson, and Jens Lagergren. Fitting points on the real line and its application to RH mapping. In *Algorithms—ESA '98 (Venice)*, volume 1461 of *Lecture Notes in Comput. Sci.*, pages 465–476. Springer, Berlin, 1998. 1.2

Piotr Indyk. Algorithmic aspects of geometric embeddings. In *Proceedings of the 42nd Symposium on the Foundations of Computer Science (FOCS)*, pages 10–33, 2001. 1

Claire Kenyon, Yuval Rabani, and Alistair Sinclair. Low distortion maps between point sets. In *Proceedings of the 36th ACM Symposium on the Theory of Computing (STOC)*, pages 272–380, 2004. 1, 1.3

Philip Klein, Ajit Agarwal, R. Ravi, and Satish Rao. Approximation through multicommodity flow. In *IEEE Symposium on Foundations of Computer Science*, pages 726–737, 1990. 3.4.1

Philip N. Klein and R. Ravi. A nearly best-possible approximation algorithm for node-weighted steiner trees. *J. Algorithms*, 19(1):104–115, 1995. 2.1.2

Jon Kleinberg, Alex Slivkins, and Tom Wexler. Triangulation and embedding using small sets of beacons. In *Proceedings of the 45th Symposium on the Foundations of Computer Science (FOCS)*, 2004. 7

Jon Kleinberg and Éva Tardos. Approximation algorithms for classification problems with pairwise relationships: metric labeling and Markov random fields. *Journal of the ACM (JACM)*, 49(5):616–639, 2002. ISSN 0004-5411. (Preliminary version in *40th FOCS*, 1999). 1

Robert Krauthgamer, James Lee, Manor Mendel, and Assaf Naor. Measured descent: A new embedding method for finite metrics. In *Proceedings of the 45th Symposium on the Foundations of Computer Science (FOCS)*, pages 434–443, 2004. 1.4, 5, 5, 28

E. L. Lawler, J. K. Lenstra, A. H. G. Rinnooy Kan, and editors D. B. Shmoys. *The Traveling Salesman Problem*. John Wiley & Sons, 1985. 1.1

James Lee, Manor Mendel, and Assaf Naor. Metric structures in $l_1$: Dimension, snowflakes, and average distortion. In *In Proceedings of LATIN*, 2004. 2.4

F. Thomson Leighton and Satish B. Rao. Multicommodity max-flow min-cut theorems and their use in designing approximation algorithms. *Journal of the ACM*, 46(6):787–832, 1999. (Preliminary version in *29th FOCS*, pages 422–431, 1988). 1.1

Nathan Linial, Eran London, and Yuri Rabinovich. The geometry of graphs and some of its algorithmic applications. *Combinatorica*, 15(2):215–245, 1995. ISSN 0209-9683. (Preliminary version in *35th FOCS*, 1994). 1, 1

Jiří Matoušek. *Lectures on discrete geometry*, volume 212 of *Graduate Texts in Mathematics*. Springer-Verlag, New York, 2002. ISBN 0-387-95373-6. 1

Jiří Matoušek. Bi-Lipschitz embeddings into low dimensional Euclidean spaces. *Commentationes Mathematicae Universitatis Carolinae*, 31(3):589–600, 1990. 1, 8

Christos H. Papadimitriou. The NP-completeness of the bandwidth minimization problem. *Computing*, 16(3):263–270, 1976. 1.4, 5

Yuri Rabinovich. On average distortion of embedding metrics into the line and into $\ell_1$. In *Proceedings of the thirty-fifth ACM symposium on Theory of computing*, pages 456–462. ACM Press, 2003. ISBN 1-58113-674-9. 1.1

Yuri Rabinovich and Ran Raz. Lower bounds on the distortion of embedding finite metric spaces in graphs. *Discrete Comput. Geom.*, 19(1):79–94, 1998. ISSN 0179-5376. 1

Satish Rao and Andrea Richa. New approximation techniques for some ordering problems. In *Proceedings of the 9th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 211–218, 1998. 1.1

James B. Saxe. Embeddability of graphs into $k$-space is strongly np-hard. In *Allerton Conference in Communication, Control and Computing*, pages 480–489, 1979. 1.2, 3

Yossi Shiloach. A minimum linear arrangement algorithm for undirected trees. *SIAM Journal on Computing*, 8(1), 1979. 4, 2.3

R.A. Sitters. The minimum latency problem is np-hard for weighted trees. In *W.J. Cook, A.S. Schulz (eds.), Integer Programming and Combinatorial Optimization, Lecture Notes in Computer Science 2337*, pages 230–239, 2002. 1, 1.1, 2.1.1

Dan Spielman and Shang-Hua Teng. Nearly-linear time algorithms for graph partitioning, graph sparsification, and solving linear systems. In *Proceedings of the 36th ACM Symposium on the Theory of Computing (STOC)*, 2004. 1.5

M. S. Waterman, T. S. Smith, M. Singh, and W. A. Beyer. Additive evolutionary trees. *Journal of Theoretical Biology*, 64:199–213, 1977. 1.2

R.T. Wong. Worst-case analysis of network design problem heuristics. *SIAM Journal Alg. Disc. Math*, 1(1):51–63, 1980. 1