Algorithms for Low-Distortion Embeddings into Arbitrary 1-Dimensional Spaces

Timothy Carpenter
Dept. of Computer Science & Engineering, The Ohio State University
Columbus, USA
carpenter.454@osu.edu

Fedor V. Fomin
Department of Informatics, University of Bergen
Norway
fomin@ii.uib.no

Daniel Lokshtanov
Department of Informatics, University of Bergen
Norway
daniello@ii.uib.no

Saket Saurabh
Institute of Mathematical Sciences
Chennai, India
saket@imsc.res.in

Anastasios Sidiropoulos
Computer Science Dept., University of Illinois at Chicago
USA
sidiropo@uic.edu

Abstract

We study the problem of finding a minimum-distortion embedding of the shortest path metric of an unweighted graph into a "simpler" metric X. Computing such an embedding (exactly or approximately) is a non-trivial task even when X is the metric induced by a path, or, equivalently, the real line. In this paper we give approximation and fixed-parameter tractable (FPT) algorithms for minimum-distortion embeddings into the metric of a subdivision of some fixed graph H, or, equivalently, into any fixed 1-dimensional simplicial complex. More precisely, we study the following problem: For given graphs G, H and integer c, is it possible to embed G with distortion c into a graph homeomorphic to H? Then embedding into the line is the special case $H = K_2$, and embedding into the cycle is the case $H = K_3$, where K_k denotes the complete graph on k vertices. For this problem we give

- an approximation algorithm, which in time $f(H)\cdot\text{poly}(n)$, for some function f, either correctly decides that there is no embedding of G with distortion c into any graph homeomorphic to H, or finds an embedding with distortion $\text{poly}(c)$;
- an exact algorithm, which in time $f'(H,c)\cdot\text{poly}(n)$, for some function f', either correctly decides that there is no embedding of G with distortion c into any graph homeomorphic to H, or finds an embedding with distortion c.

Prior to our work, poly(OPT)-approximation or FPT algorithms were known only for embedding into paths and trees of bounded degrees.

2012 ACM Subject Classification Mathematics of computing → Approximation algorithms

Keywords and phrases Metric embeddings, minimum-distortion embeddings, 1-dimensional simplicial complex, Fixed-parameter tractable algorithms, Approximation algorithms

Digital Object Identifier 10.4230/LIPIcs.SoCG.2018.21

Related Version A full version of the paper is available at https://arxiv.org/abs/1712.06747

1 Introduction

Embeddings of various metric spaces are a fundamental primitive in the design of algorithms [16, 18, 23, 22, 1, 2]. A low-distortion embedding into a low-dimensional space can be used as a sparse representation of a metrical data set (see e.g. [17]). Embeddings into 1- and 2-dimensional spaces also provide a natural abstraction of vizualization tasks (see e.g. [9]). Moreover embeddings into topologically restricted spaces can be used to discover interesting structures in a data set; for example, embedding into trees is a natural mathematical abstraction of phylogenetic reconstruction (see e.g. [11]). More generally, embedding into "algorithmically easy" spaces provides a general reduction for solving geometric optimization problems (see e.g. [7, 12]).

A natural algorithmic problem that has received a lot of attention in the past decade concerns the exact or approximate computation of embeddings of minimum distortion of a given metric space into some host space (or, more generally, into some space chosen from a specified family). Despite significant efforts, most known algorithms for this important class of problems work only for the case of the real line and trees.

In this work we present exact and approximate algorithms for computing minimum distortion embeddings into arbitrary 1-dimensional topological spaces of bounded complexity. More precisely, we obtain algorithms for embedding the shortest-path metric of a given unweighted graph into a subdivision of an arbitrary graph H. The case where H is just one edge is precisely the problem of embedding into the real line. We remark that prior to our work, even the case where H is a triangle, which corresponds to the problem of embedding into a cycle, was open.

We remark that the problem of embedding shortest path metrics of finite graphs into any fixed finite 1-dimensional simplicial complex $\mathcal{C}$ is equivalent to the problem of embedding into arbitrary subdivisions of some fixed finite graph H, where H is the abstract 1-dimensional simplicial complex corresponding to $\mathcal{C}$[1]. Since we are interested in algorithms, for the remainder of the paper we state all of our results as embeddings into subdivisions of graphs.

1.1 Our contribution

We now formally state our results and briefly highlight the key new techniques that we introduce. The input space consists of some unweighted graph G. The target space is some unknown subdivision H' of some fixed unweighted graph H; we allow the edges in H' to have arbitrary non-negative edge lengths.

We first consider the problem of approximating a minimum-distortion embedding into arbitrary H-subdivisions. We obtain a polynomial-time approximation algorithm, summarized in the following. The proof is given in Section 4.

[1] Here, a d-dimensional simplicial complex, for some integer $d \geq 1$, is the space obtained by taking a set of simplices of dimension at most d, and identifying pairs of faces of the same dimension. An abstract d-dimensional simplicial complex $\mathcal{A}$ is a family of nonempty subsets of cardinality at most $d+1$ of some ground set X, such that for all $Y' \subset Y \in \mathcal{A}$, we have $Y' \in \mathcal{A}$; in particular, any 1-dimensional simplicial complex corresponds to the set of edges and vertices of some graph.

▶ **Theorem 1.** *There exists a $8^h n^{\mathcal{O}(1)}$ time algorithm that takes as input an n-vertex graph G, a graph H on h vertices, and an integer c, and either correctly concludes that there is no c-embedding of G into any subdivision of H, or produces a c_{ALG}-embedding of G into a subdivision of H, with $c_{\mathsf{ALG}} \leq 64 \cdot 10^6 \cdot c^{24}(h+1)^9$.*

In addition, we also obtain a FPT algorithm, parameterized by the optimal distortion and H.

▶ **Theorem 2.** *There exists a $f(h,c) \cdot n^{\mathcal{O}(1)}$ time algorithm that takes as input an n-vertex graph H, a graph H on h vertices, and an integer c, and either correctly concludes that there is no c-embedding of G into any subdivision of H, or produces a c-embedding of G into a subdivision of H.*

1.2 Related work

Embedding into 1-dimensional spaces. Most of the previous work on approximation and FPT algorithms for low-distortion embedding (with one notable recent exception [27]) concerns embeddings of a more general metric space M into the real line and trees. However, even in the case of embedding into the line, all polynomial time approximation algorithms make assumptions on the metric M such as having bounded spread (which is the ratio between the maximum and the minimum point distances in M) [3, 26] or being the shortest-path metric of an unweighted graph [5]. This happens for a good reason: as it was shown by Bădoiu *et al.* [3], computing the minimum line distortion is hard to approximate up to a factor polynomial in n, even when M is a weighted tree metric with spread $n^{\mathcal{O}(1)}$.

Most relevant to our approximation algorithm is the work of Bădoiu *et al.* [5], who gave an algorithm that for a given n-vertex (unweighted) graph G and $c > 0$ in time $\mathcal{O}(cn^3)$ either concludes correctly that no c-distortion of G into line exists, or computes an $\mathcal{O}(c)$-embedding of G into the line. Similar results can be obtained for embedding into trees [5, 6]. Our approximation algorithm can be seen as an extension of these results to much more general metrics.

Parameterized complexity of low-distortion embeddings was considered by Fellows *et al.* [13], who gave an FPT algorithm for finding an embedding of an unweighted graph metric into the line with distortion at most c, or concludes that no such embedding exists, which works in time $\mathcal{O}(nc^4(2c+1)^{2c})$. As it was shown by Lokshtanov *et al.* [24], unless the exponential time hypothesis fails, this bound is asymptotically tight. For weighted graph metrics Fellows *et al.* obtained an algorithm with running time $\mathcal{O}(n(cW)^4(2c+1)^{2cW})$, where W is the largest edge weight of the input graph. In addition, they rule out, unless P=NP, any possibility of an algorithm with running time $\mathcal{O}((nW)^{h(c)})$, where h is a function of c alone. The problem of low-distortion embedding into a tree is FPT parameterized by the maximum vertex degree in the tree and the distortion c [13].

Due to the intractability of low-distortion embedding problems from the approximation and parameterized complexity perspective, Nayyeri and Raichel [26] initiated the study of approximation algorithms with running time, in the worst case, not necessarily polynomial and not even FPT. In a very recent work Nayyeri and Raichel [27] obtained a $(1+\varepsilon)$-approximation algorithm for finding the minimum-distortion embedding of an n-point metric space M into the shortest path metric space of a weighted graph H with m vertices. The running time of their algorithm is $(c_{\mathsf{OPT}}\Delta)^{\omega\cdot\lambda\cdot(1/\varepsilon)^{\lambda+2}\cdot\mathcal{O}((c_{\mathsf{OPT}})^{2\lambda})} \cdot n^{\mathcal{O}(\omega)} \cdot m^{\mathcal{O}(1)}$, where Δ is the spread of the points of M, ω is the treewidth of H and λ is the doubling dimension of H. Our approximation and FPT algorithms and the exact algorithm of Nayyeri and Raichel are incomparable. Their algorithm is for more general metrics but runs in polynomial time only when the optimal distortion c_{OPT} is constant, even when H is a cycle. In contrast, our approximation algorithm runs in polynomial time for any value of c_{OPT}. Moreover, the algorithm of Nayyeri and Raichel is (approximation) FPT with parameter c_{OPT} only when the spread Δ of M (which in the case of the unweighted graph metric is the diameter of the graph) and the doubling dimension of the host space are both constants; when $c_{\mathsf{OPT}} = \mathcal{O}(1)$ (which is the interesting case for FPT algorithms), this implies that the doubling dimension of M must also be constant, and therefore M can contain only a constant number of points, this makes the problem trivially solvable in constant time. The running time of our parameterized algorithm does not depend on the spread of the metric of M.

Embedding into higher dimensional spaces. Embeddings into d-dimensional Euclidean space have also been investigated. The problem of approximating the minimum distortion in this setting appears to be significantly harder, and most known results are lower bounds [25, 10]. Specifically, it has been shown by Matoušek and Sidiropoulos [25] that it is NP-hard to approximate the minimum distortion for embedding into $\mathbb{R}^2$ to within some polynomial factor. Moreover, for any fixed $d \geq 3$, it is NP-hard to distinguish whether the optimum distortion is at most α or at least n^{β}, for some universal constants $\alpha, \beta > 0$. The only known positive results are a $\mathcal{O}(1)$-approximation algorithm for embedding subsets of the 2-sphere into $\mathbb{R}^2$ [5], and approximation algorithms for embedding ultrametrics into $\mathbb{R}^d$ [4, 9].

Bijective embeddings. We note that the approximability of minimum-distortion embeddings has also been studied for the case of bijections [28, 15, 19, 21, 8, 20, 10]. In this setting, most known algorithms work for subsets of the real line and for trees.

2 Notation and definitions

For a graph G, we denote by $V(G)$ the set of vertices of G and by $E(G)$ the set of edges of G. For some $U \subseteq V(G)$, we denote by $G[U]$ the subgraph of G induced by U. Let $\mathsf{deg}_{\mathsf{max}}(G)$ denote the maximum degree of G.

Let $M = (X, d)$, $M' = (X', d')$ be metric spaces. An injective map $f : X \to X'$ is called an *embedding*. The *expansion* of f is defined to be $\mathsf{expansion}(f) = \sup_{x' \neq y' \in X} \frac{d'(f(x'), f(y'))}{d(x', y')}$ and the *contraction* of f is defined to be $\mathsf{contraction}(f) = \sup_{x \neq y \in X} \frac{d(x,y)}{d'(f(x), f(y))}$. We say that f is *non-expanding* (resp. *non-contracting*) if $\mathsf{expansion}(f) \leq 1$ (resp. $\mathsf{contraction}(f) \leq 1$). The distortion of f is defined to be $\mathsf{distortion}(f) = \mathsf{expansion}(f) \cdot \mathsf{contraction}(f)$. We say that f is a *c-embedding* if $\mathsf{distortion}(f) \leq c$.

For a metric space $M = (X, d)$, for some $x \in X$, and $r \geq 0$, we write $\mathsf{ball}_M(x, r) = \{y \in X : d(x, y) \leq r\}$, and for some $Y \subseteq X$, we define $\mathsf{diam}_M(Y) = \sup_{x,y \in Y} d(x, y)$. We omit the subscript when it is clear from the context. We also write $\mathsf{diam}(M) = \mathsf{diam}_M(X)$. When M is finite, the *local density* of M is defined to be $\delta(M) = \max_{x \in X, r > 0} \frac{|\mathsf{ball}_M(x,r)| - 1}{2r}$. For a graph G, we denote by d_G the shortest-path distance in G. We shall often use G to refer to the metric space $(V(G), d_G)$.

For graphs H and H', we say that H' is a *subdivision* of H if it is possible, after replacing every edge of H by some path, to obtain a graph isomorphic to H'.

3 Overview of our results and techniques

Here we present our main theorems and algorithms, with a short discussion. Formal proofs and detailed statements of the algorithms can be either found in the later section or in the appended full version of the paper.

Approximation algorithm for embedding into an H-subdivision for general H. Here, we briefly highlight the main ideas of the approximation algorithm for embedding into H-subdivisions, for arbitrary fixed graph H. A key concept is that of a *proper* embedding: this is an embedding where every edge of the target space is "necessary". In other words, for every edge e of H' there exists some vertices u, v in G, such that the shortest path between u and v in H' traverses e. Embeddings that are not proper are difficult to handle. We first guess the set of edges in H such that their corresponding paths in H' contain unnecessary edges; we "break" those edges of H into two new edges, each having a leaf as one of their endpoint. There is a bounded number of guesses (depending on H), and we are guaranteed that for at least one guess, there exists an optimal embedding that is proper. By appropriately scaling the length of the edges in H' we may assume that the embedding we are looking for has contraction exactly 1. The importance of using proper embeddings is that a proper embedding which is "locally" non-contracting is also (globally) non-contracting, while this is not necessarily true for non-proper embeddings.

A second difficulty is that we do not know the number of times that an edge in H is being subdivided. Guessing the exact number of times each edge is subdivided would require $n^{f(H)}$ time, which is too much. Instead we set a specific threshold ℓ, based on c. The threshold ℓ is approximately c^3, and essentially ℓ is a threshold for how many vertices a BFS in G needs to see before it is able to distinguish between a part of G that is embedded on an edge, and a part of G that is embedded in an area of H' close to a vertex of degree at least 3. In particular, parts of G that are embedded close to the middle of an edge *can* be embedded with low distortion onto the line, while parts that are embedded close to a vertex of degree 3 in H can not - because G "grows in at least 3 different directions" in such parts. Since a BFS can be used as an approximation algorithm for embedding into the line, it will detect whether the considered part of G is close to a degree ≥ 3 vertex of H or not.

Instead of guessing exactly how many times each edge of H is subdivided, we guess for every edge whether it is subdivided at least ℓ times or not. The edges of H that are subdivided at least ℓ times are called "long", while the edges that are subdivided less than ℓ times are called "short". We call the connected components of H induced by the short edges a *cluster*. Having defined clusters, we now observe that a cluster with only two long edges leaving it *can* be embedded into the line with (relatively) low distortion, contradicting what we said in the previous paragraph! Indeed, the parts of G mapped to a cluster with only two long edges leaving it are (from the perspective of a BFS), indistinguishable from the parts that are mapped in the middle of an edge! For this reason, we classify clusters into two types: the *boring* ones that have at most two (long) edges leaving them, and the *interesting* ones that are incident to at least 3 long edges.

Any graph can be partitioned into vertices of degree at least 3 and paths between these vertices such that every internal vertex on these paths has degree 2. Thinking of clusters as "large" vertices and the long edges as edges between clusters, we can now partition the "cluster graph" into interesting clusters (analogs of the vertices of degree at least 3), and chains of boring clusters between the interesting clusters (analogs of the paths of vertices of degree 2).

The parts of G that are embedded onto a chain of boring clusters can be embedded into the line with low distortion, and therefore, for a BFS these parts are indistinguishable from the parts of G that are embedded onto a single long edge. However, the interesting clusters are distinguishable from the boring ones, and from the parts of G that are mapped onto long edges, because around interesting clusters the graph really does "grow in at least 3 different directions" for a long enough time for a BFS to pick up on this.

Using the insights above, we can find a set F of at most $|V(H)|$ vertices in G, such that every vertex in F is mapped "close" to some interesting cluster, and such that every interesting cluster has some vertex in F mapped "close" to it. At this point, one can essentially just guess in time $\mathcal{O}(h^h)$ which vertex of F is mapped close to which clusters of H. Then one maps each of the vertices that are "close" to F (in G) to some arbitrarily chosen spot in H which is close enough to the image of the corresponding vertex of F. Local density arguments show that there are not too many vertices in G that are "close" to F, and therefore this arbitrary choice will not drive the distortion of the computed mapping up too much.

It remains to embed all of the vertices that are "far" from F in G. However, by the choice of F we know that all such vertices should be embedded onto long edges, or onto chains of boring clusters. Thus, each of the yet un-embedded parts of the graph can be embedded with low distortion into the line! All that remains is to compute such low distortion embeddings for each part using a BFS, and assign each part to an edge of H. Stitching all of these embeddings together yields the approximation algorithm.

There are multiple important details that we have completely ignored in the above exposition. The most important one is that a cluster can actually be quite large when compared to a long edge. After all, a boring cluster contains up to $E(H)$ short edges, and the longest short edge can be almost as long as the shortest long edge! This creates several technical complications in the algorithm that computes the set F. Resolving these technical complications ends up making it unnecessary to guess which vertex of F is mapped to which vertex of H, instead one can compute this directly, at the cost of increasing the approximation ratio.

FPT algorithm for embedding into an H-subdivision for general H. Our FPT algorithm for embedding a graph G into H-subdivisions (for arbitrary fixed H) draws inspiration from the algorithm for the line used in [14, 5], while also using an approach similar to the approximation algorithm for H-subdivisions. The result here is an exact algorithm with running time $f(H, c_{\mathsf{OPT}}) \cdot n^{\mathcal{O}(1)}$. A naive generalization of the algorithm for the line needs to maintain the partial solution over $f(H)$ intervals, which results in running time $n^{g(H)}$, which is too much. Supposing that there is a proper c-embedding of G into some H-subdivision, we attempt to find this embedding by guessing the short and long edges of H. Using this guess, we partition H into connected clusters of short and long edges (we call the clusters of short edges "interesting" clusters, and the clusters of long edges "path" clusters). We show that if a c-embedding exists, we can find a subset of $V(G)$, with size bounded by a function of $|H|$ and c, that contains all vertices embedded into the interesting clusters of H. From this, we make further guesses as to which specific vertices are embedded into which interesting clusters, then how they are embedded into the interesting clusters. We also make guesses as to what the embedding looks like for a short distance (for example, $\mathcal{O}(c^2)$) along the long edges which are connected to the important clusters.

Since the number of guesses at each step so far can be bounded in terms of c and H, we can iterate over all possible configurations. Once our guesses have found the correct choices for the interesting clusters and for a short distance along the paths leaving these clusters, we are able to partition the remaining vertices of G, and guess which path clusters these partitions are embedded into. Due to the "path-like" nature of the path clusters, when we pair the correct partition and path cluster, we are able to use an approach inspired by [14, 5] to find a c-embedding of the partition into the path cluster, which is compatible with the choices already made for the interesting clusters.

4 An approximation algorithm for embedding into arbitrary graphs

In this section we give a complete (technical) description of our approximation algorithm for embedding into arbitrary graphs; albeit without proof sometimes. In particular, we prove Theorem 1. We start by formally defining the notions of pushing and proper embeddings.

4.1 Proper, pushing, and non-contracting embeddings

Let G, H be connected graphs, with a fixed total order $<$ on $V(G)$ and $V(H)$. A non-contracting, c_{OPT}-embedding of G to H is a function $f_{\mathsf{OPT}} : V(G) \to (H_{\mathsf{OPT}}, w_{\mathsf{OPT}})$, where H_{OPT} is a subdivision of H, $w_{\mathsf{OPT}} : E(H_{\mathsf{OPT}}) \to \mathbb{R}^{>0}$, and for all $u, v \in V(G)$,

$$d_G(u,v) \le d_{(H_{\mathsf{OPT}},w_{\mathsf{OPT}})}(f_{\mathsf{OPT}}(u), f_{\mathsf{OPT}}(v)) \le c_{\mathsf{OPT}} \cdot d_G(u,v),$$

where $d_{(H_{\mathsf{OPT}},w_{\mathsf{OPT}})}$ is the shortest path distance in H_{OPT} with respect to w_{OPT}. Stated formally, for all $h_1, h_2 \in V(H_{\mathsf{OPT}})$, if $\mathcal{P}$ is the set of all paths from h_1 to h_2 in H_{OPT}, then

$$d_{(H_{\mathsf{OPT}},w_{\mathsf{OPT}})}(h_1,h_2) = \min_{P \in \mathcal{P}} \left\{ \sum_{e \in P} w_{\mathsf{OPT}}(e) \right\}.$$

▶ **Definition 3.** For a graph G_1, a subdivision G_1' of G_1, and and edge $e \in E(G_1)$, let $\mathsf{SUB}_{G_1'}(e)$ be the subdivision of e in G_1'. For convenience, for each $e \in E(H)$, we shall use e_{OPT} to indicate the subdivision of e in H_{OPT}.

The following notion of consecutive vertices will be necessary to describe additional properties we will want our embeddings to have.

▶ **Definition 4.** Suppose there exists $u, v \in V(G)$ and $e \in E(H)$ such that $f_{\mathsf{OPT}}(u), f_{\mathsf{OPT}}(v) \in V(e_{\mathsf{OPT}})$ and $f_{\mathsf{OPT}}(u) < f_{\mathsf{OPT}}(v)$. If for all $w \in V(G) \setminus \{u, v\}$, $f_{\mathsf{OPT}}(w)$ is not in the path in e_{OPT} between $f_{\mathsf{OPT}}(u)$ and $f_{\mathsf{OPT}}(v)$, then we say that u and v are *consecutive w.r.t.* e, or we say that u and v are *consecutive*.

The first property we will want our embeddings to have is that they are "pushing". The intuition here is that we want our embedding to be such that we cannot modify it by contracting the distance further between two consecutive vertices.

▶ **Definition 5.** If for all $u, v \in V(G)$ and $e \in E(H)$ such that u and v are consecutive w.r.t. e we have that $d_{e_{\mathsf{OPT}}}(f_{\mathsf{OPT}}(u), f_{\mathsf{OPT}}(v)) = d_G(u,v)$, then we say that f_{OPT} is *pushing*.

The next property we want for our embeddings is that they are "proper", meaning that all edges of the target graph are, in a loose sense, covered by an edge of the source graph.

▶ **Definition 6.** For any $z \in V(H_{\mathsf{OPT}})$, if there exists $\{u, v\} \in E(G)$ such that

$$d_{(H_{\mathsf{OPT}},w_{\mathsf{OPT}})}(f_{\mathsf{OPT}}(u), f_{\mathsf{OPT}}(v)) = d_{(H_{\mathsf{OPT}},w_{\mathsf{OPT}})}(z, f_{\mathsf{OPT}}(u)) + d_{(H_{\mathsf{OPT}},w_{\mathsf{OPT}})}(z, f_{\mathsf{OPT}}(v))$$

then we say that z is *proper w.r.t.* f_{OPT}. If for all $x \in V(H_{\mathsf{OPT}})$, x is proper w.r.t. f_{OPT}, then we say that f_{OPT} is *proper*.

Given some target graph to embed into, there may not necessarily be a proper embedding. However, for some "quasi-subgraph" (defined below) of our target, there will be a proper embedding, which can be used to find an embedding into the target graph.

▶ **Definition 7.** Let J and J' be connected graphs. We say J' is a *quasi-subgraph* of J if J can be made isomorphic to J' by applying any sequence of the following rules to J: (1.) Delete a vertex in $V(J)$. (2.) Delete an edge in $E(J)$. (3.) Delete an edge $\{u, v\} \in E(J)$, add vertices u', v' to $V(J)$, and add edges $\{u, u'\}, \{v, v'\}$ to $E(J)$.

By examining the quasi-subgraphs of our target graph, we can restrict our search to proper, pushing, non-contracting embeddings. This leads to the following result.

▶ **Lemma 8.** *There exists a proper, pushing, non-contracting c_{OPT}-embedding of G to some (H^q, w^q), where H^q is the subdivision of some quasi-subgraph of H, and $w^q : E(H^q) \to \mathbb{R}^{>0}$.*

4.2 Approximation algorithm

In this subsection we give our approximation algorithm. By Lemma 8 there is a proper, pushing c_{OPT}-embedding of G into a subdivision of a quasi-subgraph H^q of H with edge weight function w^q. Furthermore, by subdividing each edge of H sufficiently many times, for any $\epsilon > 0$ any c-embedding of G into (H^q, w^q) can be turned into an $(c + \epsilon)$-embedding of G into a subdivision of H.

The weighted quasi-subgraph (H^q, w^q) of H is a subdivision of a quasi-subgraph H_{sub} of H. Since H has less than $3^{|E(H)|+|V(H)|}$ quasi-subgraphs our algorithm can guess H_{sub}. Thus, for the purposes of our approximation algorithm, it is sufficient to find an embedding of G into a weighted subdivision (H_{ALG}, w_{ALG}) of H_{sub} under the assumption that a proper, pushing c_{OPT}-embedding of G into some weighted subdivision of H_{sub} exists. Furthermore, any proper and pushing embedding is non-contracting and has contraction exactly equal to 1. Such an embedding f is a c-embedding if and only if for every edge

$$uv \in E(G), d_{(H,w)}(f(u), f(v)) \leq c. \tag{1}$$

Thus, to prove that our output embedding is a c-embedding (for some c) we will prove that it is proper, pushing and that (1) is satisfied. Thus, the main technical result of this section is encapsulated in the following lemma.

▶ **Lemma 9.** *There is an algorithm that takes as input a graph G with n vertices, a graph H and an integer c, runs in time $2^h \cdot n^{\mathcal{O}(1)}$ and either correctly concludes that there is no c-embedding of G into a weighted subdivision of H, or produces a proper, pushing c_{ALG}-embedding of G into a weighted subdivision of a quasi-subgraph H' of H, where $c_{\mathsf{ALG}} = \mathcal{O}(c^{24}h^9)$.*

Definitions. To prove Lemma 9 we need a few definitions. Throughout the section we will assume that there exists a weighted subdivision $(H_{\mathsf{OPT}}, w_{\mathsf{OPT}})$ and a c-embedding $f_{\mathsf{OPT}} : V(G) \to V(H_{\mathsf{OPT}})$. This embedding is unknown to the algorithm and will be used for analysis purposes only. Every edge $e = uv$ in H corresponds to a path P_e in H_{OPT} from u to v. Based on the embedding $f_{\mathsf{OPT}} : V(G) \to V(H_{\mathsf{OPT}})$ we define the *embedding pattern* function $\hat{f}_{\mathsf{OPT}} : V(G) \to V(H) \cup E(H)$ as follows. For every vertex $v \in V(G)$ such that f_{OPT} maps v to a vertex of H_{OPT} that is also a vertex of H, $\hat{f}_{\mathsf{OPT}}$ maps v to the same vertex. In other words if $f_{\mathsf{OPT}}(v) = u$ for $u \in V(H)$, then $\hat{f}_{\mathsf{OPT}}(v) = u$. Otherwise f_{OPT} maps v to a vertex u on a path P_e corresponding to an edge $e \in E(H)$. In this case we set $\hat{f}_{\mathsf{OPT}}(v) = e$.

We will freely make use of the "inverses" of the functions f_{OPT} and $\hat{f}_{\mathsf{OPT}}$. For a vertex set $C \subseteq V(H_{\mathsf{OPT}})$ we define $f_{\mathsf{OPT}}^{-1}(C) = \{v \in V(G) : f_{\mathsf{OPT}}(v) \in C\}$. We will also naturally extend functions that act on elements of a universe to subsets of that universe. For example, for a set $F \subseteq E(H_{\mathsf{OPT}})$ we use $w_{\mathsf{OPT}}(F)$ to denote $\sum_{e \in F} w_{\mathsf{OPT}}(e)$. We further extend this convention to write $w_{\mathsf{OPT}}(P_e)$ instead of $w_{\mathsf{OPT}}(E(P_e))$ for a path (or a subgraph) of H_{OPT}. We extend the distance function to also work for distances between sets.

Throughout the section we will use the following parameters, for now ignore the parenthesized comments to the definitions of the parameters, these are useful for remembering the purpose of the parameter when reading the proofs: $h = |E(H)|$ (the number of edges in H), c (the distortion of f_{OPT}), $\ell = 20c^3$ (long edge threshold), $r = 5\ell h$ (half of covering radius), and $c_{ALG} = 64 \cdot 10^6 \cdot c^{24}(h+1)^9$ (distortion of output embedding).

Using the parameter ℓ we classify the edges of H into short and long edges. An edge $e \in E(H)$ is called *short* if $w_{\mathsf{OPT}}(P_e) \leq \ell$ and it is called *long* otherwise. The edge sets E_{short} and E_{long} denote the set of short and long edges in H respectively. A *cluster* in H is a connected component C of the graph $H_{short} = (V(H), E_{short})$. We abuse notation and denote by C both the connected component and its vertex set. The *long edge degree* of a cluster C in H is the number of long edges in H incident to vertices in C. Here a long edge whose both endpoints are in C is counted twice. A cluster C of long edge degree at most 2 is called *boring*, otherwise it is *interesting*. Most of the time when discussing clusters, we will be speaking of clusters in H. However we overload the meaning of the word cluster to mean something else for vertex sets of G. A *cluster in* G is a set C such that there exists a cluster C_H of H such that $C = \{v \in V(G) \ : \ \hat{f}_{\mathsf{OPT}}(v) \in V(C_H) \cup E(C_H)\}$. Thus there is a one to one correspondence between clusters in G and H.

A *cluster-chain* is a sequence $C_1, e_1, C_2, e_2, \ldots, e_{t-1}, C_t$ such that the following conditions are satisfied. First, the C_i's are distinct clusters in H, except that possibly $C_1 = C_t$. Second, C_1 and C_t are interesting, while $C_2, \ldots C_{t-1}$ are boring. Finally, for every $i < t$ the edge e_i is a long edge in H connecting a vertex of C_i to a vertex of C_{i+1}.

4.2.1 Using Breadth First Search to detect interesting clusters

In this subsection we prove a lemma that is the main engine behind Lemma 9. Once the main engine is set up, all we will need to finish the proof of Lemma 9 will be to complete the embedding by running the approximation algorithm for embedding into a line for each cluster-chain of H, and stitching these embeddings together.

Before stating the lemma we define what it means for a vertex set F in G to cover a cluster. We say that a vertex set $F \subseteq V(G)$ r-*covers* a cluster C in G if some vertex in F is at distance at most r (in G) from at least one vertex in C. A vertex set $F \subseteq V(G)$ covers a cluster C in H if F covers the cluster C_G corresponding to C in G.

▶ **Lemma 10** (Interesting Cluster Covering Lemma). *There exists an algorithm that takes as input G, H and c, runs in time $2^h n^{\mathcal{O}(1)}$ and halts. If there exists a proper c-embedding $\hat{f}_{\mathsf{OPT}}$ from G to a weighted subdivision of H, the algorithm outputs a family $\mathcal{F} \subset 2^{V(G)}$ such that $|\mathcal{F}| \leq 2^h$, every set $F \in \mathcal{F}$ has size at most h, and there exists an $F \in \mathcal{F}$ that $2r$-covers all interesting clusters of H.*

We do not prove Lemma 10 here, however we describe the algorithm used in Lemma 10. The algorithm iteratively adds vertices to a set F. During the iteration the algorithm makes some non-deterministic steps, these steps result in the algorithm returning a family of sets $\mathcal{F}$ rather than a single set F. We now describe a crucial subroutine of the algorithm of Lemma 10 that we call the SEARCH algorithm. The algorithm takes as input G, c, a set $F \subseteq V(G)$ and a vertex v. The algorithm explores the graph, starting from v with the aim of finding a local structure in G that on one hand can not be embedded into the line with low distortion, while on the other hand is far away from F. It will either output fail, meaning that the algorithm failed to find a structure not embeddable into the line, or $\mathsf{success}$ together with a vertex $\hat{u}$, meaning that the algorithm succeeded to find a structure not embeddable into the line, and that $\hat{u}$ is close to this structure.

Description of the SEARCH algorithm. The algorithm takes as input G, c, a set $F \subseteq V(G)$ and a vertex v. It performs a breadth first search (BFS) from v in G. Let X_1, X_2, etc. be the BFS layers starting from v. In other words $X_i = \{x \in V(G) \ : \ d_G(v,x) = i\}$. The algorithm inspects the BFS layers $X_1, X_2, \ldots$ one by one in increasing order of i.

For $i < 2c^2$ the algorithm does nothing other than the BFS itself. For $i = 2c^2$ the algorithm proceeds as follows. It picks an arbitrary vertex $v_L \in X_i$ and picks another vertex $v_R \in X_i$ at distance at least $2c+1$ from v_L in G. Such a vertex v_R might not exist, in this case the algorithm proceeds without picking v_R. The algorithm partitions X_i into X_i^L and X_i^R in the following way. For every vertex $x \in X_i$, if $d_G(x, v_L) \leq 2c$ then x is put into X_i^L. If $d_G(x, v_R) \leq 2c$ then x is put into X_i^R. If some vertex $x \in X_i$ is put *both* into X_i^L and in X_i^R, or *neither* into X_i^L nor into X_i^R the algorithm returns $\mathsf{success}$ together with $\hat{u} = v$.

For $i > 2c^2$ the algorithm proceeds as follows. If any vertex in X_i is at distance at most r from any vertex in F (in the graph G), the algorithm outputs fail and halts. Otherwise, the algorithm partitions X_i into X_i^L and X_i^R. A vertex $x \in X_i$ is put into X_i^L if x has a neighbor in X_{i-1}^L and into X_i^R if x has a neighbor in X_{i-1}^R. Note that x has at least one neighbor in X_{i-1}, and so x will be put into at least one of the sets X_i^L or X_i^R. If x is put into both sets X_i^L and X_i^R, the algorithm outputs $\mathsf{success}$ with $\hat{u} = x$ and halts. If $|X_i^L| > 2c^2$ or if two vertices in X_i^L have distance at least $2c+1$ from each other in G the algorithm picks a vertex $x \in X_i^L$ and returns $\mathsf{success}$ with $\hat{u} = x$. Similarly, if $|X_i^R| > 2c^2$ or if two vertices in X_i^R have distance at least $2c+1$ from each other in G the algorithm picks a vertex $x \in X_i^R$ and returns $\mathsf{success}$ with $\hat{u} = x$. If the BFS stops, (i.e $X_i = \emptyset$), the algorithm outputs fail.

4.2.2 STITCHing together approximate line embeddings

We now describe the STITCH algorithm. This algorithm takes as input G, H, c and $F \subseteq V(G)$, runs in polynomial time and halts. We will prove that if there exists a c-embedding f_{OPT} of G into a weighted subdivision $(H_{\mathsf{OPT}}, w_{\mathsf{OPT}})$ of H such that F $2r$-covers all interesting clusters of G, the algorithm produces a c_{ALG}-embedding f_{ALG} of G into a weighted subdivision $(H_{\mathsf{ALG}}, w_{\mathsf{ALG}})$ of a quasi-subgraph H' of H. Throughout this section we will assume that such an embedding f_{OPT} exists.

The STITCH algorithm starts by setting $R = 4r$, $\Delta = 4r$ and then proceeds as follows. As long as there are two vertices u and v in F such that $2R \leq d_G(u,v) \leq 2R + \Delta$, the algorithm increases R to $R + \Delta$. Note that this process will stop after at most $\binom{|F|}{2}$ iterations, and therefore when it terminates we have $R \leq 4r \cdot h^2 \leq 400c^3h^3$. Define $B = \mathsf{ball}_G(F, R)$, and $\mathcal{B}$ to be the family of connected components of $G[B]$. Notice that the previous process ensures that for any $B_1, B_2 \in \mathcal{B}$ we have $d_G(B_1, B_2) \geq \Delta$. Notice further that for every interesting cluster C in H we have that $\mathsf{ball}_G(\hat{f}_{\mathsf{OPT}}^{-1}(C), r) \subseteq B$.

We now classify the connected components of $G - B$. A component Z of $G - B$ is called *deep* if it contains at least one vertex at distance (in G) at least $\frac{\Delta}{2}$ from F, and it is *shallow* otherwise. The shallow components are easy to handle because they only contain vertices close to F.

▶ **Lemma 11.** *For every shallow component Z of $G - B$, there is at most one connected component $B_1 \in \mathcal{B}$ that contains neighbors of Z.*

The next sequence of lemmas allows us to handle deep components. We say that a component Z in $G - B$ *lies on* the cluster-chain $\chi = C_1, e_1, \ldots, C_t$ if

$$Z \subseteq \left(\bigcup_{i \leq t} \hat{f}_{\mathsf{OPT}}^{-1}(C_i) \cup \hat{f}_{\mathsf{OPT}}^{-1}(e_i) \right) \setminus \hat{f}_{\mathsf{OPT}}^{-1}(C_1 \cup C_t).$$

▶ **Lemma 12.** *Every component Z of $G - B$ lies on some cluster-chain and no two deep components Z_1, Z_2 of $G - B$ can lie on the same cluster-chain χ.*

▶ **Lemma 13.** *There is a polynomial time algorithm that given G, B and a component Z of $G - B$ computes an embedding of Z components of $G - B$ into the line with distortion at most $(\ell \cdot h \cdot c)^4$. Furthermore, all vertices in Z with neighbors outside Z are mapped by this embedding within distance $(\ell \cdot h \cdot c)^6$ from the end-points.*

Proof. Let Z be a component of $G - B$. By Lemma 12, Z lies on a cluster-chain $\chi = C_1, e_1, \dots, C_t$. Define the following total ordering of the vertices in Z: If $\hat{f}_{\mathsf{OPT}}(a) \in C_i \cup \{e_i\}$ and $\hat{f}_{\mathsf{OPT}}(b) \in C_j \cup \{e_j\}$ and $i < j$, then a comes before b. If $\hat{f}_{\mathsf{OPT}}(a) \in C_i$ and $\hat{f}_{\mathsf{OPT}}(b) = e_i$ then a comes before b. If $\hat{f}_{\mathsf{OPT}}(a) = \hat{f}_{\mathsf{OPT}}(b) = e_i$ and $f_{\mathsf{OPT}}(a)$ is closer than $f_{\mathsf{OPT}}(b)$ to C_{i-1}, then a comes before b. If $\hat{f}_{\mathsf{OPT}}(a) \in C_i$ and $\hat{f}_{\mathsf{OPT}}(b) \in C_i$ we place a and b in any relative order.

At most $\ell \cdot h$ vertices are mapped to any boring cluster C_i, and the distance between any two vertices in the same boring cluster in H_{OPT} is at most $\ell \cdot h$. Thus the distance (in G) between any two consecutive vertices in this ordering is at most $\ell \cdot h \cdot c$. The number of vertices appearing in the ordering between the two endpoints of an edge is at most $\ell \cdot h$ (all the vertices of a boring cluster). Thus, if the ordering is turned into a pushing, non-contracting embedding into the line, the distortion of this embedding is at most $(\ell \cdot h)^2 \cdot c$. Using the known polynomial time approximation algorithm for embedding into the line [5] we can find an embedding of Z into the line with distortion at most $(\ell \cdot h \cdot c)^4$ in polynomial time.

Because B is a union of at most h balls, it follows that at most $c^2 \cdot h^2$ vertices in Z have neighbors in G, and that all of these vertices are among the $\ell \cdot h$ first or last ones in the above ordering. Since any two low distortion embeddings of a metric space into the line map the same vertices close to the end-points, it follows that all vertices in Z with neighbors outside Z are mapped by this embedding within distance $(\ell \cdot h \cdot c)^6$ from the end-points. ◀

The STITCH algorithm builds the graph H' as follows. Every vertex of H' corresponds to a connected component $B \in \mathcal{B}$. Every deep component Z of $G - B$ corresponds to an edge between the (at most two) sets B_1 and $B_2 \in \mathcal{B}$ that have non-empty intersection with $N_G(Z)$. Note that the graph H' is a multi-graph because it may have multiple edges and self loops. However, since each set $B \in \mathcal{B}$ has a connected image in H under $\hat{f}$, Lemma 12 implies that H' is a topological subgraph of H. Hence any weighted subdivision of H' is a weighted subdivision of a subgraph of H. The STITCH algorithm uses Lemma 13 to compute embeddings of each deep connected component Z of $G \setminus B$. Further, for each component $B_i \in \mathcal{B}$ the algorithm computes the set $B_i^\star$ which contains B_i, as well as the vertex sets of all shallow connected components whose neighborhood is in B_i. By Lemma 11 the $B_i^\star$'s together with the deep components of $G - B$ form a partition of $V(G)$.

What we would like to do is to map each set $B_i^\star$ onto the vertex of H' that it corresponds to, and map each deep connected component Z of $G - B$ onto the edge of H' that it corresponds to. When mapping Z onto the edge of H we use the computed embedding of Z into the line, and subdivide this edge appropriately.

The reason this does not work directly is that we may not map all the vertices of $B_i^\star$ onto the single vertex v_i in H' that corresponds to B_i. Instead, STITCH picks one of the edges incident to v_i, sub-divides the edge an appropriate number of times, and maps all the vertices of $B_i^\star$ onto the newly created vertices on this edge. The order in which the vertices of $B_i^\star$ are mapped onto the edge is chosen arbitrarily, however all of these vertices are mapped closer to v_i than any vertices of the deep component Z that is mapped onto the edge. This concludes the construction of H_{ALG} and f_{ALG}. The STITCH algorithm defines a weight function w_{ALG} on the edges of H_{ALG}, such that the embedding is pushing and non-contracting.

▶ **Lemma 14.** *f_{ALG} is a c_{ALG}-embedding of G into $(H_{\mathsf{ALG}}, w_{\mathsf{ALG}})$.*

We are now ready to prove Lemma 9.

Proof of Lemma 9. The algorithm runs the algorithm of Lemma 10, to produce a collection $\mathcal{F}$, such that $|\mathcal{F}| \leq 2^h$, every set in $\mathcal{F}$ has size at most h, and such that if G has a c-embedding f_{OPT} of into a weighted subdivision of H, then some $F \in \mathcal{F}$ $2r$-covers all interesting clusters (of f_{OPT}) in G. For each $F \in \mathcal{F}$ the algorithm runs the STITCH algorithm, which takes polynomial time. If STITCH outputs a c_{ALG}-embedding of G into a weighted subdivision of a quasi-subgraph H' of H, the algorithm returns this embedding.

By Lemma 14, for the choice of $F \in \mathcal{F}$ that $2r$-covers all interesting clusters, the STITCH algorithm does output a c_{ALG}-embedding of G into a weighted subdivision of a quasi-subgraph H' of H. This concludes the proof. ◀

The discussion prior to the statement of Lemma 9 immediately implies that Lemma 9 is sufficient to give an approximation algorithm for finding a low distortion (not necessarily pushing, proper or non-contracting) embedding G into a (unweighted) subdivision of H. The only overhead of the algorithm is the guessing of the quasi-subgraph H_{sub} of H, this incurs an additional factor of $3^{|V(H)|+|E(H)|} \leq 9^h$ in the running time, yielding the proof of Theorem 1.

Finally, we remark that at a cost of a potentially higher running time in terms of h, one may replace the $(h+1)^9$ factor with c^9. If $c \geq h+1$ we have that $c_{\mathsf{ALG}} \leq 64 \cdot 10^6 \cdot c^{33}$. On the other hand, if $c \leq h+1$ we may run the algorithm of Theorem 2 in time $f(H)n^{\mathcal{O}(1)}$ instead and solve the problem optimally.

References

1 Sanjeev Arora, James Lee, and Assaf Naor. Euclidean distortion and the sparsest cut. *Journal of the American Mathematical Society*, 21(1):1–21, 2008.

2 Sanjeev Arora, Satish Rao, and Umesh Vazirani. Expander flows, geometric embeddings and graph partitioning. *Journal of the ACM (JACM)*, 56(2):5, 2009.

3 Mihai Bădoiu, Julia Chuzhoy, Piotr Indyk, and Anastasios Sidiropoulos. Low-distortion embeddings of general metrics into the line. In *Proceedings of the 37th Annual ACM Symposium on Theory of Computing (STOC)*, pages 225–233. ACM, 2005.

4 Mihai Bădoiu, Julia Chuzhoy, Piotr Indyk, and Anastasios Sidiropoulos. Embedding ultrametrics into low-dimensional spaces. In *Proceedings of the 22nd Annual Symposium on Computational Geometry (SoCG)*, pages 187–196. ACM, 2006.

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

6 Mihai Bădoiu, Piotr Indyk, and Anastasios Sidiropoulos. Approximation algorithms for embedding general metrics into trees. In *Proceedings of the 18th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 512–521. ACM and SIAM, 2007.

7 Yair Bartal. Probabilistic approximation of metric spaces and its algorithmic applications. In *Foundations of Computer Science, 1996. Proceedings., 37th Annual Symposium on Foundations of Computer Science*, pages 184–193. IEEE, 1996.

8 Nishanth Chandran, Ryan Moriarty, Rafail Ostrovsky, Omkant Pandey, Mohammad Ali Safari, and Amit Sahai. Improved algorithms for optimal embeddings. *ACM Transactions on Algorithms*, 4(4), 2008. `doi:10.1145/1383369.1383376`.

9 Mark de Berg, Krzysztof Onak, and Anastasios Sidiropoulos. Fat polygonal partitions with applications to visualization and embeddings. *Journal of Computational Geometry*, 4(1):212–239, 2013.

10 Jeff Edmonds, Anastasios Sidiropoulos, and Anastasios Zouzias. Inapproximability for planar embedding problems. In *Proceedings of the 20th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 222–235. Society for Industrial and Applied Mathematics, 2010.

11 Martin Farach, Sampath Kannan, and Tandy Warnow. A robust model for finding optimal evolutionary trees. *Algorithmica*, 13(1-2):155–179, 1995.

12 Martin Farach-Colton and Piotr Indyk. Approximate nearest neighbor algorithms for hausdorff metrics via embeddings. In *Proceedings of the 40th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 171–179. IEEE, 1999.

13 Michael Fellows, Fedor V. Fomin, Daniel Lokshtanov, Elena Losievskaja, Frances Rosamond, and Saket Saurabh. Distortion is fixed parameter tractable. *ACM Trans. Comput. Theory*, 5(4):16:1–16:20, 2013. `doi:10.1145/2489789`.

14 Michael R. Fellows, Fedor V. Fomin, Daniel Lokshtanov, Elena Losievskaja, Frances A. Rosamond, and Saket Saurabh. Distortion is fixed parameter tractable. In *Proceedings of the 36th International Colloquium on Automata, Languages and Programming (ICALP)*, volume 5555 of *Lecture Notes in Computer Science*, pages 463–474. Springer, 2009.

15 Alexander Hall and Christos Papadimitriou. Approximating the distortion. In Chandra Chekuri, Klaus Jansen, José D. P. Rolim, and Luca Trevisan, editors, *Approximation, Randomization and Combinatorial Optimization. Algorithms and Techniques*, pages 111–122, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

16 Piotr Indyk. Algorithmic applications of low-distortion geometric embeddings. In *Proceedings of the 42nd IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 10–33. IEEE, 2001.

17 Piotr Indyk. Stable distributions, pseudorandom generators, embeddings, and data stream computation. *Journal of the ACM (JACM)*, 53(3):307–323, 2006.

18 Piotr Indyk and Jiri Matousek. Low-distortion embeddings of finite metric spaces. In *Handbook of Discrete and Computational Geometry*, pages 177–196. CRC Press, 2004.

19 Claire Kenyon, Yuval Rabani, and Alistair Sinclair. Low distortion maps between point sets. In *Proceedings of the 36th Annual ACM Symposium on Theory of Computing (STOC)*, pages 272–280. ACM, 2004.

20 Claire Kenyon, Yuval Rabani, and Alistair Sinclair. Low distortion maps between point sets. *SIAM J. Comput.*, 39(4):1617–1636, 2009. `doi:10.1137/080712921`.

21 Subhash Khot and Rishi Saket. Hardness of embedding metric spaces of equal size. In *Approximation, randomization, and combinatorial optimization. Algorithms and techniques*, pages 218–227. Springer, 2007.

22 Nathan Linial. Finite metric-spaces—combinatorics, geometry and algorithms. In *Proceedings of the International Congress of Mathematicians, Vol. III*, pages 573–586, Beijing, 2002. Higher Ed. Press.

23 Nathan Linial, Eran London, and Yuri Rabinovich. The geometry of graphs and some of its algorithmic applications. *Combinatorica*, 15(2):215–245, 1995.

24 Daniel Lokshtanov, Dániel Marx, and Saket Saurabh. Slightly superexponential parameterized problems. In *Proceedings of the 21st Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 760–776. SIAM, 2011.

25 Jiří Matoušek and Anastasios Sidiropoulos. Inapproximability for metric embeddings into $\mathbb{R}^d$. *Transactions of the American Mathematical Society*, 362(12):6341–6365, 2010.

26 Amir Nayyeri and Benjamin Raichel. Reality distortion: Exact and approximate algorithms for embedding into the line. In *Proceedings of the 56th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 729–747. IEEE, 2015.

27 Amir Nayyeri and Benjamin Raichel. A treehouse with custom windows: Minimum distortion embeddings into bounded treewidth graphs. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '17, pages 724–736, Philadelphia, PA, USA, 2017. Society for Industrial and Applied Mathematics. URL: `http://dl.acm.org/citation.cfm?id=3039686.3039732`.

28 Christos Papadimitriou and Shmuel Safra. The complexity of low-distortion embeddings between point sets. In *Proceedings of the 16th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, volume 5, pages 112–118. SIAM, 2005.