Low-distortion Embeddings of General Metrics Into the Line*

Mihai Bădoiu
MIT

Julia Chuzhoy
MIT

Piotr Indyk
MIT

Anastasios Sidiropoulos †
MIT

ABSTRACT

A low-distortion embedding between two metric spaces is a mapping which preserves the distances between each pair of points, up to a small factor called distortion. Low-distortion embeddings have recently found numerous applications in computer science.

Most of the known embedding results are "absolute", that is, of the form: any metric Y from a given class of metrics C can be embedded into a metric X with low distortion c. This is beneficial if one can guarantee low distortion for all metrics Y in C. However, in many situations, the worst-case distortion is too large to be meaningful. For example, if X is a line metric, then even very simple metrics (an n-point star or an n-point cycle) are embeddable into X only with distortion linear in n. Nevertheless, embeddings into the line (or into low-dimensional spaces) are important for many applications.

A solution to this issue is to consider "relative" (or "approximation") embedding problems, where the goal is to design an (a-approximation) algorithm which, given any metric X from C as an input, finds an embedding of X into Y which has distortion $a * c_Y(X)$, where $c_Y(X)$ is the best possible distortion of an embedding of X into Y.

In this paper we show algorithms and hardness results for relative embedding problems. In particular we give:

- an algorithm that, given a general metric M, finds an embedding with distortion $O(\Delta^{3/4}\text{poly}(c_{line}(M)))$, where Δ is the spread of M
- an algorithm that, given a weighted tree metric M, finds an embedding with distortion $\text{poly}(c_{line}(M))$
- a hardness result, showing that computing minimum line distortion is hard to approximate up to a factor polynomial in n, even for weighted tree metrics with spread $\Delta = n^{O(1)}$.

Categories and Subject Descriptors

F.2 [**Theory of Computation**]: ANALYSIS OF ALGORITHMS AND PROBLEM COMPLEXITY

General Terms

Algorithms

Keywords

Metric embeddings, approximation algorithms

1. INTRODUCTION

A low-distortion embedding between two metric spaces with distance functions D and D' is a (non-contractive) mapping f such that for any pair of points p, q in the original metric, their distance $D(p, q)$ before the mapping is the same as the distance $D'(f(p), f(q))$ after the mapping, up to a (small) multiplicative factor c. Low-distortion embeddings have been a subject of extensive mathematical studies. More recently, they found numerous applications in computer science (cf. [15, 13]).

Most of the research on embeddings focused on showing *absolute* results, of the form:

> Given a class of metrics C and a metric Y, what is the *smallest distortion* $c \geq 1$ such that any metric $X \in C$ can be embedded into Y with distortion c ?

Very recently, a few papers addressed the *relative* [1] (or approximation) version of the problem, which is of the following form:

> Given a class of metrics C and a metric Y, what is the *smallest approximation factor* $a \geq 1$ of a polynomial-time algorithm minimizing the distortion of embedding of a given input metric $X \in C$ into Y ?

*This research was supported in part by David and Lucille Packard Fellowship and Alfred P. Sloan Fellowship.
†Supported in part by Alexandros S. Onassis Public Benefit Foundation and NSF CAREER grant CCR-0133849.

STOC'05, May 22-24, 2005, Baltimore, Maryland, USA.

[1]The *absolute* and *relative* (resp.) versions of the problem were referred to as *combinatorial* and *algorithmic* (resp.) in [4]. These terms could be confusing, however, since the *absolute* problem has both combinatorial and algorithmic components: in many applications it is important how to find low-distortion embeddings, in addition to knowing that such embeddings exist. Thus, to avoid misunderstanding, in this paper we use a different terminology.

Paper	From	Into	Distortion	Comments
[16]	general metrics	l_2	c	uses SDP
[14]	line	line	c	c is constant, embedding is a bijection
	unweighted graphs	bounded degree trees	c	c is constant, embedding is a bijection
[19]	$\mathbb{R}^3$	$\mathbb{R}^3$	$> (3-\epsilon)c$	hard to 3-approximate, embedding is a bijection
[9]	unweighted graphs	sub-trees	$O(c \log n)$	
[5]	unweighted graphs	trees	$O(c)$	
[4]	unweighted graphs	line	$O(c^2)$	implies $\sqrt{n}$-approximation
			$> ac$	Hard to a-approximate for some $a > 1$
			c	c is constant
	unweighted trees	line	$O(c^{3/2}\sqrt{\log c})$	
	subsets of a sphere	plane	$3c$	

Figure 1: Previous work on relative embedding problems for multiplicative distortion.

From	Into	Distortion	Comments
general metrics	line	$O(\Delta^{3/4}c^{11/4})$	
weighted trees	line	$c^{O(1)}$	
weighted trees	line	$\Omega(n^{1/12}c)$	Hard to $O(n^{1/12})$-approximate even for $\Delta = n^{O(1)}$

Figure 2: Our results.

The relative formulation is of interest in situations where the absolute formulation yields distortion that is too large to be interesting or meaningful. A good example is the problem of embedding metrics into a line. Even simple metrics, such as an n-point star or an n-point cycle requires $\Omega(n)$ distortion when embedded into a line. Nevertheless, line embeddings, or, in general, embeddings into low-dimensional spaces, are important in many applications, such as visualisation (e.g., see [20] or [18] web pages). Thus, it is important to design algorithms which produce low-distortion embeddings, if such embeddings are possible.

Despite the importance of the problem, not many relative embedding results are known. This is perhaps because the problems do not seem to be easily amenable[2] to standard approximation algorithms approaches (which were, e.g., successfully used for a closely related *bandwidth* problem [11, 8]). The results that we are aware of[3] are listed in Figure 1 (c denotes the optimal distortion, and n denotes metric size).

In this paper, we consider the problem of embedding metrics induced by *weighted* graphs into the line. The known algorithms were designed for *unweighted graphs* and thus provide only very weak guarantees for the problem. Specifically, assume that the minimum interpoint distance between the points is 1 and the maximum distance[4] is Δ. Then, by scaling, one can obtain algorithms for weighted graphs, with approximation factor multiplied by Δ.

Our results are presented in Figure 2. The first result is an algorithm that, given a general metric c-embeddable into the line, constructs an embedding with distortion $O(\Delta^{3/4}c^{11/4})$. The algorithm uses a novel method for traversing a weighted graph. It also uses a modification of the unweighted-graph algorithm from [4] as a subroutine, with a more general analysis.

Then, we consider the problem of embedding weighted tree metrics into the line. In this case we are able to get rid of the dependence on Δ from the approximation factor. Specifically, our algorithm produces an embedding with distortion $c^{O(1)}$.

We complement our upper bounds by a lower bound, which shows that the problem is hard to approximate up to a factor $a = \Omega(n^{1/12})$. This dramatically improves over the earlier result of [4], which only showed that the problem is hard for some constant $a > 1$ (note however that their result applies to unweighted graph metrics as well). Since the instances used to show our hardness result have spread $\Delta \le n^{O(1)}$, it follows that approximating the distortion up to a factor of $\Delta^{\Omega(1)}$ is hard as well. In fact, the instances used to show hardness are metrics induced by (weighted) *trees*; thus the problem is hard for tree metrics as well. Our hardness proof is inspired by the ideas of Unger [21].

1.1 Related Work

Relative embedding problems have been theoretically studied for over a decade. Until recently, however, the research has been mostly focused on different notions of distortion. Specifically, several results gave been obtained for finding embedding f from space (X, D) into (X', D') that minimizes the *maximum additive distortion*, that is, minimizing $\max_{p,q \in X} |D(p,q) - D'(f(p), f(q))|$. The results are depicted in Figure 3. A few other results have been obtained for *average* distortion [6, 7]; see the papers for results and problem definitions.

[2]For example, there exist metrics for which any vertex ordering resulting in "low" bandwidth must result in "high" distortion when converted into a (non-contractive) embedding. This holds, e.g., for a metric induced by a "comb" graph, with a "teeth", each of length b, for $b >> a$. The row-by-row order, which minimizes the bandwidth, results in $\Omega(ab)$ distortion of the edges at the end of the teeth, while the column-by-column order gives distortion b.

[3]Note that the table contains only the results that hold for the *multiplicative* definition of the distortion. There is a rich body of work that applies to other definitions of distortion, notably the *additive* or *average* distortion, summarized in Section 1.1.

[4]We call the maximum/minimum interpoint distance ratio the *spread* of the metric.

Paper	From	Into	Distortion	Comments
[10]	general distance matrix	ultrametrics	c	
[1]	general distance matrix	tree metrics	$3c$	
			$\geq 9/8c$	Hard to 9/8-approximate
[12]	general distance matrix	line	$2c$	
			$\geq 4/3c$	Hard to 4/3-approximate
[2]	general distance matrix	plane under l_1	$O(c)$	
[3]	general distance matrix	plane under l_2	$O(c)$	Time quasi-polynomial in Δ

Figure 3: Previous work on relative embedding problems for maximum additive distortion.

2. PRELIMINARIES

Consider an embedding of a set of vertices V into the line. We say that $U \subset V$ is embedded *continuously*, if there are no vertices $x, x' \in U$, and $y \in V - U$, such that $f(x) < f(y) < f(x')$.

We say that vertex set U is embedded *inside* vertex set U' iff the smallest interval containing the embedding of U also contains the embedding of U'. In particular, we say that vertex v is embedded inside edge $e = (x, y)$ for $v \neq x, v \neq y$, if either $f(x) < f(v) < f(y)$ or $f(y) < f(v) < f(x)$ hold.

Let $M = (X, D)$ be a metric, and $f : X \to \mathbb{R}$ be a non-contracting embedding of M into the line. Then, the *length* of f is $\max_{u \in X} f(u) - \min_{v \in X} f(v)$.

3. GENERAL METRICS

In this section we will present a polynomial-time algorithm that given a metric $M = (X, D)$ of spread Δ that c-embeds into the line, computes an embedding of M into the line, with distortion $O(c^{11/4}\Delta^{3/4})$. Since it is known [17] that any n-point metric embeds into the line with distortion $O(n)$, we can assume that $\Delta = O(n^{4/3})$.

We view metric M as a complete graph G defined on vertex set X, where the weight of each edge $e = \{u, v\}$ is $D(u, v)$. As a first step, our algorithm partitions the point set X into sub-sets $X_1, \ldots, X_\ell$, as follows. Let W be a large integer to be specified later. Remove all the edges of weight greater than W from G, and denote the resulting connected components by $C_1, \ldots, C_\ell$. Then for each $i : 1 \leq i \leq \ell$, X_i is the set of vertices of C_i. Let G_i be the subgraph of G induced by X_i. Our algorithm computes a low-distortion embedding for each G_i separately, and then concatenates the embeddings to obtain the final embedding of M. In order for the concatenation to have small distortion, we need the length of the embedding of each component to be sufficiently small (relatively to W). The following simple lemma, essentially shown in [17], gives an embedding that will be used as a subroutine.

Lemma 1. *Let $M = (X, D)$ be a metric with minimum distance 1, and let T be a spanning tree of M. Then we can compute in polynomial time an embedding of M into the line, with distortion $O(cost(T))$, and length $O(cost(T))$.*

The embedding in the lemma is computed by taking an (pre-order) walk of the tree T. Since each edge is traversed only a constant number of times, the total length and distortion of the embedding follows.

Our algorithm proceeds as follows. For each $i : 1 \leq i \leq \ell$, we compute a spanning tree T_i of G_i, that has the following properties: the cost of T_i is low, and there exists a walk on T_i that gives a small distortion embedding of G_i. We can then view the concatenation of the embeddings of the components as if it is obtained by a walk on a spanning tree T of G. We show that the cost of T is small, and thus the total length of the embedding of G is also small. Since the minimum distance between components is large, the inter-component distortion is small.

3.1 Embedding the Components

In this section we concentrate on some component G_i, and we show how to embed it into a line.

Let H be the graph on vertex set X_i, obtained by removing all the edges of length at least W from G_i, and let H' be the graph obtained by removing all the edges of length at least cW from G_i. For any pair of vertices $x, y \in X_i$, let $D_H(x, y)$ and $D_{H'}(x, y)$ be the shortest-path distances between x and y in H and H', respectively. Recall that by the definition of X_i, H is a connected graph, and observe that $D_H(x, y) \geq D_{H'}(x, y) \geq D(x, y)$.

Lemma 2. *For any $x, y \in X_i$, $D_{H'}(x, y) \leq cD(x, y)$.*

Proof: Let f be an optimal non-contracting embedding of G_i, with distortion at most c. Consider any pair u, v of vertices that are embedded consecutively in f. We start by showing that $D(u, v) \leq cW$. Let T be the minimum spanning tree of H. If edge $\{u, v\}$ belongs to T, then $D(u, v) \leq W$. Otherwise, since T is connected, there is an edge $e = \{u', v'\}$ in tree T, such that both u and v are embedded inside e. But then $D(u', v') \leq W$, and since the embedding distortion is at most c, $|f(u) - f(v)| \leq |f(u') - f(v')| \leq cW$. As the embedding is non-contracting, $D(u, v) \leq cW$ must hold.

Consider now some pair $x, y \in X_i$ of vertices. If no vertex is embedded between x and y, then by the above argument, $D(x, y) \leq cW$, and thus the edge $\{x, y\}$ is in H' and $D_{H'}(x, y) = D(x, y)$. Otherwise, let $z_1, \ldots, z_k$ be the vertices appearing in the embedding f between x and y (in this order). Then the edges $\{x, z_1\}$, $\{z_1, z_2\}$, $\ldots$, $\{z_{k-1}, z_k\}$, $\{z_k, y\}$ all belong to H', and therefore

$$\begin{aligned}
& D_{H'}(x, y) \\
\leq\ & D_{H'}(x, z_1) + D_{H'}(z_1, z_2) + \ldots D_{H'}(z_{k-1}, z_k) + D_{H'}(z_k, y) \\
=\ & D(x, z_1) + D(z_1, z_2) + \ldots D(z_{k-1}, z_k) + D(z_k, y) \\
\leq\ & |f(x) - f(z_1)| + |f(z_1) - f(z_2)| + \ldots + |f(z_{k-1}) - f(z_k)| \\
& + |f(z_k) - f(y)| \\
=\ & |f(x) - f(y)| \leq cD(x, y)
\end{aligned}$$

□

We can now concentrate on embedding graph H'. Since the weight of each edge in graph H' is bounded by $O(cW)$, we can use a modified version of the algorithm of [4] to embed each G_i. First, we need the following technical Claim.

Claim 1. *For any $u, u' \in X_i$, there exists a shortest path $p = v_1, \ldots, v_k$, from u to u' in H', such that for any i, j, with $|i-j| > 1$, $D(v_i, v_j) = \Omega(W|i-j|)$.*

Proof: Pick an arbitrary shortest path, and repeat the following: while there exist consecutive vertices x_1, x_2, x_3 in p, with $D_{H'}(x_1, x_3) < cW$, remove x_2 from p, and add the edge $\{x_1, x_3\}$ in p. □

The algorithm works as follows. We start with the graph H', and we guess points u, u', such that there exists an optimal embedding of G_i having u and u' as the left-most and right-most point respectively. Let $p = (v_1, \ldots, v_k)$ be the shortest path from u to u' on H' (here $v_1 = u$ and $v_k = u'$), that is given by Claim 1. We partition X_i into clusters $V_1, \ldots, V_k$, as follows. Each vertex $x \in X_i$ belongs to cluster V_j, that minimizes $D(x, v_j)$.

Our next step is constructing super-clusters $U_1, \ldots, U_s$, where the partition induced by $\{V_j\}_{j=1}^k$ is a refinement of the partition induced by $\{U_j\}_{j=1}^s$, such that there is a small-cost spanning tree T' of G_i that "respects" the partition induced by $\{U_j\}_{j=1}^s$. More precisely, each edge of T' is either contained in a super-cluster U_i, or it is an edge of the path p. The final embedding of G_i is obtained by a walk on T', that traverses the super-clusters $U_1, \ldots, U_s$ in this order.

Note that there exist metrics over G_i for which any spanning tree that "respects" the partition induced by V_j's is much more expensive that the minimum spanning tree. Thus, we cannot simply use $U_j = V_j$.

We now show how to construct the super-clusters $U_1, \ldots, U_s$. We first need the following three technical claims, which constitute a natural extensions of similar claims from [4] to the weighted case. Their proofs are given in Appendix, Section A.

Claim 2. *For each $i : 1 \leq i \leq k$, $\max_{u \in V_i} \{D(u, v_i)\} \leq c^2 W/2$.*

Claim 3. *For each $r \geq 1$, and for each $i : 1 \leq i \leq k - r + 1$, $\sum_{j=i}^{i+r-1} |V_i| \leq c^2 W(c + r - 1) + 1$.*

Claim 4. *If $\{x, y\} \in E(H')$, where $x \in V_i$, and $y \in V_j$, then $D(v_i, v_j) \leq cW + c^2 W$, and $|i - j| = O(c^2)$.*

Let α be an integer with $0 \leq \alpha < c^4 W$. We partition the set X_i into super-clusters $U_1, \ldots, U_s$, such that for each $l : 1 \leq l \leq s$, U_l is the union of $c^4 W$ consecutive clusters V_j, where the indexes j are shifted by α. We refer to the above partition as α-*shifted*.

Claim 5. *Let T be an MST of G_i. We can compute in polynomial time a spanning tree T' of G_i, with cost$(T') = O($cost$(T))$, and an α-shifted partition of X_i, such that for any edge $\{x, y\}$ of T', either both $x, y \in U_l$ for some $l : 1 \leq l \leq s$, or $x = v_j$ and $y = v_{j+1}$ for some $j : 1 \leq j < k$.*

Proof: Observe that since H is connected, all the edges of T can have length at most W, and thus T is a subgraph of both H and H'. Consider the α-shifted partition obtained by picking $\alpha \in \{0, \ldots, c^4 W - 1\}$, uniformly at random. Let T' be the spanning tree obtained from T as follows: For all edges $\{x, y\}$ of T, such that $x \in V_i \subseteq U_{i'}$, and $y \in V_j \subseteq U_{j'}$, where $i' \neq j'$, we remove $\{x, y\}$ from T, and we add the edges $\{x, v_i\}$, $\{y, v_j\}$, and the edges on the subpath of p from v_i to v_j. Finally, if the resulting graph T' contains cycles, we remove edges in an arbitrary order, until T' becomes a tree. Note that although T' is a spanning tree of G_i, it is not necessarily a subtree of H'.

Clearly, since the edges $\{x, v_i\}$, and $\{y, v_j\}$ that we add at each iteration of the above procedure are contained in the sets $U_{i'}$, and $U_{j'}$ respectively, it follows that T' satisfies the condition of the Claim.

We will next show that the expectation of cost(T'), taken over the random choice of α, is $O($cost$(T))$. For any edge $\{x, y\}$ that we remove from T, the cost of T' is increased by the sum of $D(x, v_i)$ and $D(y, v_j)$, plus the length of the shortest path from v_i to v_j in H'. Observe that the total increase of cost(T') due to the subpaths of p that we add, is at most cost(T). Thus, it suffices to bound the increase of cost(T') due to the edges $\{x, v_i\}$, and $\{y, v_j\}$.

By Claim 2, $D(x, v_i) \leq c^2 W/2$, and $D(y, v_j) \leq c^2 W/2$. Thus, for each edge $\{x, y\}$ that we remove from T, the cost of the resulting T' is increased by at most $O(c^2 W)$.

For each i, the set $U_i \cup U_{i+1}$ contains $\Omega(c^4 W)$ consecutive clusters V_j. Also, by Claim 4 the difference between the indexes of the clusters V_{t_1}, V_{t_2} containing the endpoints of an edge, is at most $|t_1 - t_2| = O(c^2)$. Thus, the probability that an edge of T is removed, is at most $O(\frac{1}{c^2 W})$, and the expected total cost of the edges in $E(T') \setminus E(T)$ is $O(|X_i|) = O($cost$(T))$. Therefore, the expectation of cost(T'), is at most $O($cost$(T))$. The Claim follows by the linearity of expectation, and by the fact that there are only few choices for α. □

Let $U_1, \ldots, U_s$ be an α-shifted partition, satisfying the conditions of Claim 5, and let T' be the corresponding tree. Clearly, the subgraph $T'[U_i]$ induced by each U_i is a connected subtree of T'. For each U_i, we construct an embedding into the line by applying Lemma 1 on the spanning tree $T'[U_i]$. By Claim 3, $|U_i| = O(c^6 W^2)$, and by Claim 2, the cost of the spanning tree $T'[U_i]$ of U_i is at most $O(|U_i| c^2 W) = O(c^8 W^3)$. Therefore, the embedding of each U_i, given by Lemma 1 has distortion $O(c^8 W^3)$, and length $O(c^8 W^3)$.

Finally, we construct an embedding for G_i by concatenating the embeddings computed for the sets $U_1, U_2, \ldots, U_s$, while leaving sufficient space between each consecutive pair of super-clusters, so that we satisfy non-contraction.

Lemma 3. *The above algorithm produces a non-contracting embedding of G_i with distortion $O(c^8 W^3)$ and length $O(cost(MST(G_i)))$.*

Proof: Let g be the embedding produced by the algorithm. Clearly, g is non-contracting. Consider now a a pair of points $x, y \in X$, such that $x \in U_i$, and $y \in U_j$. If $|i - j| \leq 1$, then $|g(x) - g(y)| = O(c^8 W^3)$, and thus the distortion of $D(x, y)$ is at most $O(c^8 W^3)$.

Assume now that $|i - j| \geq 2$, and $x \in V_{i'}$, $y \in V_{j'}$. Then $|g(x) - g(y)| = O(|i - j| \cdot c^8 W^3)$. On the other hand, $D(x, y) \geq D(v_{i'}, v_{j'}) - D(v_{i'}, x) - D(v_{j'}, y) \geq D(v_{i'}, v_{j'}) - c^2 W \geq D_{H'}(v_{i'}, v_{j'})/c - c^2 W \geq |i' - j'|/c - c^2 W = \Omega(|i - j| c^4 W^2)$. Thus, the distortion on $\{x, y\}$ is $O(c^4 W)$. In total, the maximum distortion of the embedding g is $O(c^8 W^3)$.

In order to bound the length of the constructed embedding, consider a walk on T' that visits the vertices of T according to their appearance in the line, from left to right. It is easy to see that this walk traverses each edge at most 4 times. Thus, the length of the embedding, which is equal to the total length of the walk is at most $4\text{cost}(T') = O(\text{cost}(T))$. $\square$

3.2 The Final Embedding

We are now ready to give a detailed description of the final algorithm. Assume that the minimum distance in M is 1, and the diameter is Δ. Let $H = (X, E)$ be a graph, such that an edge $(u, v) \in E$ iff $D(u, v) \le W$, for a threshold W, to be determined later. We use the algorithm presented above to embed every connected component $G_1, \ldots, G_k$ of H. Let $f_1, f_2, \ldots, f_k$ be the embeddings that we get for the components $G_1, G_2, \ldots G_k$ using the above algorithm, and let T be a minimum spanning tree of G. It is easy to see that T connects the components G_i using exactly $k - 1$ edges.[5] We compute our final embedding f as follows. Fix an arbitrary Eulerian walk of T. Let P be the permutation of $(G_1, G_2, \ldots, G_k)$ that corresponds to the order of the first occurrence of any node of G_i in our traversal. Compute embedding f by concatenating the embeddings f_i of components G_i in the order of this permutation. Let T_i be the minimum spanning tree of G_i. Between every 2 consecutive embeddings in the permutation f_i and f_j, leave space $\max_{u \in G_i, v \in G_j} \{D(u, v)\} = D(a, b) + O(\text{cost}(T_i)) + O(cost(T_j))$, where $D(a, b)$ is the smallest distance between components G_i and G_j. This implies the following.

Lemma 4. *The length of f is at most $O(c\Delta)$.*

Proof: The length of f is the sum of the lengths of all f_i and the space that we leave between every 2 consecutive f_i, f_j's. Then, by Lemma 3, the length of f_i is $O(c \cdot \text{cost}(T_i))$. Thus, the sum of the lengths of all f_i's is $O(c \cdot \text{cost}(T))$. The total space that we leave between all pairs of consecutive embeddings f_i is $\text{cost}(T) + 2\sum_{i=1}^{k} O(\text{cost}(T_i)) = O(\text{cost}(T))$. Therefore the total length of the embedding f is $O(\text{cost}(T))$. At the same time, the cost of T is at most the length of the optimal embedding f, which is $O(c\Delta)$. The statement follows. $\square$

Lemma 5. *Let $a \in G_i, b \in G_j$ for $i \neq j$. Then $W \le D(a, b) \le |f(a) - f(b)| \le O(c\Delta) \le O(cD(a, b)\frac{\Delta}{W})$*

Proof: The first part $D(a, b) \le |f(a) - f(b)|$ is trivial by construction, since we left enough space between components G_i and G_j. Since a and b are in difference connected components, we have $D(a, b) > W$. Using Lemma 4 we have that $|f(a) - f(b)| = O(c\Delta) = O(c\Delta\frac{D(a,b)}{W}) = O(cD(a, b)\frac{\Delta}{W})$. $\square$

Theorem 1. *Let $M = (X, D)$ be a metric with spread Δ, that embeds into the line with distortion c. Then, we can compute in polynomial time an embedding of M into the line, of distortion $O(c^{11/4}\Delta^{3/4})$.*

Proof: Consider any pair of points. If they belong to different components, their distance distortion is $O(c\Delta/W)$ (Lemma 5). If they belong to the same component, their distance distortion is $O(c^8 W^3)$ (Lemma 3). Setting $W = \Delta^{1/4}c^{-7/4}$ gives the claimed distortion bound. $\square$

[5]Follows from correctness of Kruskal's algorithm. These $k - 1$ edges are exactly the last edges to be added because they are bigger than W and within components we have edges smaller than W

4. HARDNESS OF EMBEDDING INTO THE LINE

In this section we show that even the problem of embedding weighted trees into the line is n^β-hard to approximate, for some constant $0 < \beta < 1$. Our reduction is performed from the 3SAT(5) problem, defined as follows. The input is a CNF formula φ, in which each clause consists of exactly 3 different literals and each variable participates in exactly 5 clauses, and the goal is to determine whether φ is satisfiable. Let $x_1, \ldots, x_n$, and $C_1, \ldots, C_m$, be the variables and the clauses of φ respectively, with $m = 5n/3$. Given an input formula φ, we construct a weighted tree G, such that if φ is satisfiable then there is an embedding of G into the line with distortion $O(b)$ (for some $b = \text{poly}(n)$) and if φ is not satisfiable, then the distortion of any embedding is at least $b\tau$, where $\tau = \text{poly}(n)$. The construction size is polynomial in τ, and hence the hardness result follows.

4.1 The construction

Our construction makes use of *caterpillar* graphs. A caterpillar graph consists of a path called *body*, and a collection of vertex disjoint paths, called *hairs*, while each hair is attached to a distinct vertex of the body, called the *base* of the hair. One of the endpoints of the caterpillar body is called the *first vertex* of the caterpillar, and the other endpoint is called the *last vertex.* We use two integer paremeters $b = \text{poly}(n)$ and $\tau = \text{poly}(n)$, whose exact value is determined later. We call a caterpillar graph a *canonical caterpillar*, if: (1) its body consists of integer-length edges, (2) the length of each hair is a multiple of b, and (3) each hair consists of edges of length $\frac{1}{b\tau}$. Our weighted tree G is a collection of canonical caterpillars, connected together in some way specified later. Notice that in any embedding of a canonical caterpillar with distortion less than $b\tau$, each hair must be embedded continuously (the formal proof appears below). Let $B_1, \ldots, B_t$ be caterpillars. A *concatenation* of $B_1, \ldots, B_t$ is a caterpillar obtained by connecting each pair of consecutive caterpillars B_i, B_{i+1} for $1 \le i < t$ with a unit-length edge between the last vertex of B_i and the first vertex of B_{i+1}.

The building blocks of our graph G are literal caterpillars, variable caterpillars and clause caterpillars, that represent the literals, the variables and the clauses of the input formula φ. All these caterpillars are canonical. Let x_i be some variable in formula φ. We define two caterpillars called *literal caterpillars* w_i and w'_i, which represent the literals x_i and $\overline{x}_i$, respectively. Additionally, we have a *variable* caterpillar v_i representing variable x_i.

Let Y_L and Y_R be caterpillars whose bodies contain only one vertex (denoted by L and R respectively), with a hair of length $\tau^3 b$ (denoted by H_L and H_R respectively) attached to the body. The main part of our graph G is a canonical caterpillar W, defined as a concatenation of $Y_L, w_1, w'_1, w_2, w'_2, \ldots w_n, w'_n, Y_R$. The hairs of H_L and H_R are used as padding, to ensure that all the vertices of $G \setminus (H_L \cup H_R)$, are embedded between L and R. The length of the body of W is denoted by N, and is calculated later. Variable caterpillars v_i attach to W as follows. The first vertex of v_i connects by a unit-length edge to the first vertex of w'_i.

For every clause C_j in formula φ, our construction contains a canonical caterpillar k_j representing it, which is also called a *key*. Each key k_j is attached to vertex L by an edge of length N. Figure 4 summarizes the above described

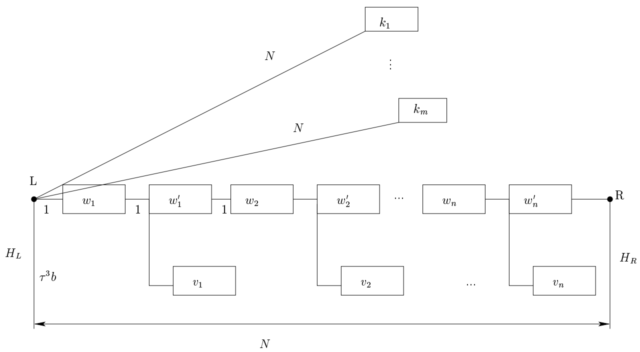


Figure 4: The high-level view of the construction.

construction.

We now provide the details on the structure of the literal caterpillars. Consider a literal ℓ, and let w be the caterpillar that represents it (i.e., if ℓ is x_i or $\overline{x}_i$, then w is w_i or $\overline{w}_i$). Assume that ℓ participates in (at most 5) clauses $C_1^\ell, C_2^\ell, \ldots$. Then w is the concatenation of at most 5 caterpillars, denoted by $h_1^\ell, h_2^\ell, \ldots$, that represent the participation of ℓ in these clauses (see Figure 5). Following [21], we call these caterpillars *keyholes*. For convenience, we ensure that for each literal ℓ there are exactly 5 such keyholes $h_1^\ell, h_2^\ell, \ldots, h_5^\ell$, as follows. If the literal participates in less than 5 clauses, we use several copies of the same keyhole that corresponds to some clause in which ℓ participates. Thus, for each clause, for each literal participating in this clause, there is at least one keyhole. All the keyholes that correspond to the same clause C_j are copies of the same caterpillar $h(j)$, called *the keyhole* of C_j.

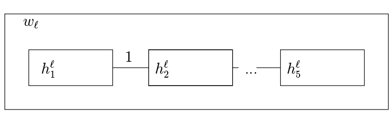


Figure 5: Caterpillar representing literal ℓ.

The main idea of the construction is as follows. First, the keys and the keyholes are designed in a special way, such that in order to avoid the distortion of $b\tau$, each key k_j has to be embedded inside one of the matching keyholes (copies of $h(j)$). The variable caterpillars are shaped in such a way that in any embedding with distortion less than $b\tau$, each variable caterpillar v_i is either embedded in w_i or w_i'. If v_i is embedded in w_i, then no key can be embedded inside any keyhole belonging to w_i without incurring the distortion of $b\tau$, and the same is true in case v_i is embedded into w_i'. Suppose formula φ is satisfiable. Then embedding of G with distortion $O(b)$ is obtained as follows. We first embed hair H_L (starting from the vertex furthest from L), then the body of W and then H_R (starting from the vertex closest to R). For each variable x_i, if the correct assignment to x_i is TRUE, then variable caterpillar v_i is embedded inside the literal caterpillar w_i', and otherwise it is embedded inside w_i. Given a clause C_j, if ℓ is the satisfied literal in this clause, we embed the key k_j in the copy of keyhole $h(j)$, that corresponds to literal ℓ. On the other hand, if φ is not satisfiable, we still need to embed each variable caterpillar v_i inside one of the two corresponding caterpillars w_i, w_i', thus defining an assignment to all the variables. For example, if v_i is embedded inside w_i, this corresponds to the assignment FALSE to variable x_i. Such embedding of v_i will block all the keyholes in the caterpillar w_i. Since the assignment is non-satisfying, for at least one of the keys k_j, all the corresponding keyholes (copies of $h(j)$) are blocked, and so in order to embed k_j, we will need to incur a distortion of $b\tau$.

The rest of the construction description, including the implementation of keys and keyholes and variable caterpillars, as well as the reduction analysis, appears in the full version of this paper.

5. APPROXIMATION ALGORITHM FOR WEIGHTED TREES

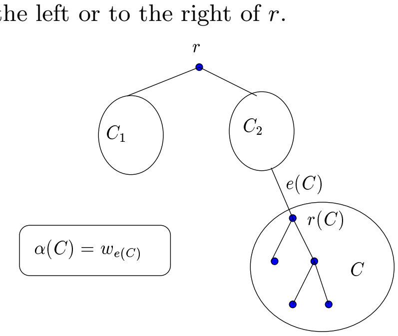
In this section we consider embedding of weighted trees into the line. Given a weighted tree T, let φ be its optimal embedding into the line, whose distortion is denoted by c (we assume that $c \geq 200$). We provide a poly(c)-approximation algorithm, which, combined with earlier work, implies $n^{1-\epsilon}$ approximation algorithm for weighted trees, for some constant $0 < \epsilon < 1$. The first step of our algorithm is guessing the optimal distortion c, and from now on we assume that we have guessed its value correctly.

We start with notation. Fix any vertex r of the tree to be the root. Given a vertex $v \neq r$, denote $d(v) = D(v,r)$. Consider any edge $e = (u,v)$. The length of e is denoted by w_e, and $d_e = \min\{d(u), d(v)\}$ is the distance of e from r. We say that e is a *large* edge if $w_e \geq \frac{d_e}{c}$, it is a *medium* edge if $\frac{d_e}{c} > w_e \geq \frac{d_e}{c^2}$, and otherwise e is a *small* edge.

CLAIM 6. *If $e = (u,v)$ is a medium or a small edge, then r is not embedded between u and v in the optimal solution.*

Proof: Assume otherwise. Then $|\varphi(u) - \varphi(v)| \geq d_e$. But $D(u,v) = w_e < \frac{d_e}{c}$, and edge e is stretched by a factor greater than c. $\square$

Let $\mathcal{C}$ be the collection of connected components, obtained by removing all the large edges from the graph. For each component $C \in \mathcal{C}$, let $r(C)$ denote its "root", i.e. the vertex of C closest to r in tree T. We also denote by $e(C)$ the unique edge incident on $r(C)$ on the path from $r(C)$ to r, and by $\alpha(C)$ the length of this edge. Clearly, in the optimal solution, the embedding of component C lies completely to the left or to the right of r.



Given some component $C \in \mathcal{C}$, let $\ell(C)$ be the vertex in C that maximizes $D(r(C), \ell(C))$, and let $P(C)$ be the path between $r(C)$ and $\ell(C)$ in tree T. We define the *radius* of C to be $s(C) = D(r(C), \ell(C))$. Component C is called *large* if $s(C) > c^4\alpha(C)$, otherwise the component is called *small*. We define a tree T' of components, whose vertex set is $\mathcal{C} \cup \{r\}$, and the edges connecting the components are the same as in the original graph, (i.e., $e(C)$ for all $C \in \mathcal{C}$.)

Procedure Partition

Let $\mathcal{C}$ be the current set of all the components.
While there is a large component $C \in \mathcal{C}$, with a medium-sized edge e on the path from $r(C)$ to $\ell(C)$, such that the removal of e splits C into two large components, do:
Let C' and C'' be the two large components obtained by removing e. Remove C from $\mathcal{C}$ and add C' and C'' to $\mathcal{C}$.

The main idea of our algorithm is to find the embedding of each one of the components separately recursively, and then concatenate these embeddings in some carefully chosen order. However, there is a problem with this algorithm, which is illustrated by the following example. Consider a large component C, consisting of a very long path, and a small component C' attached to this path in the middle. In this case any small-distortion embedding has to interleave the vertices of C and C', and thus our algorithm fails. We note that as $e(C')$ is a large edge, vertices of component C' have to be embedded into medium-sized edges of C (formal proof of this fact is provided later). In order to solve the above problem, we perform Procedure Partition, that further subdivides large components by removing some medium-size edges from them.

From now on we only consider the components after the application of the above procedure, and the component graph, the values $r(C), \ell(C), \alpha(C)$ and so on are defined with respect to these components. It is easy to see that if a medium size edge e is incident on some component C, then C is a large component.

In fact, it is more convenient for us to define and solve a slightly more general problem. In the modified problem, in addition to a weighted tree T, we are also given a threshold value H. Given any embedding of our tree into the line, we say that it satisfies the *root condition* if: (1) each component C is embedded completely to the right or to the left of r, and (2) no component C with $\alpha(C)+s(C) \geq cH$ is embedded to the right of r. Our goal is to find an embedding that satisfies the root condition, while minimizing its distortion. Even though the problem might look artificial at this point, it is easy to see that by setting $H=\infty$, it converts to our original problem. The reason for defining the problem this way is that our algorithm solves the problem recursively on each component $C \in \mathcal{C}$, and then concatenates their embeddings into the final solution. In order to avoid large distortion of the distance between r and $r(C)$, we need to impose the root condition on the sub-problem corresponding to C with threshold $H = D(r, r(C))$. We later claim that for each sub-problem there is an optimal embedding with distortion c that satisfies the corresponding root condition.

5.1 The Structure of the Optimal Solution

In this section we explore some structural properties of the optimal solution, on which our algorithm relies.

Definition 1. *Let C, C' be two large components. We say that these components are* incompatible *if $s(C) > 2c^3\alpha(C')$ and $s(C') > 2c^3\alpha(C)$.*

Lemma 6. *If C and C' are large incompatible components, then in the optimal solution they are embedded on different sides of r.*

Definition 2. *Let C be a large component, and C' a small component. We say that there is a* conflict *between C and C' iff $2c^4\alpha(C) < \alpha(C') < s(C)/2c^4$.*

Lemma 7. *If C is a large component having a conflict with small component C', then C and C' are embedded on different sides of r in the optimal solution.*

Claim 7. *Let C, C' be large components and C'' a small component. Moreover, assume that there is a conflict between C and C'' and there is a conflict between C' and C''. Then C and C' are incompatible.*

Proof: Since there is a conflict between C and C'', $\alpha(C'') > 2c^4\alpha(C)$. A conflict between C' and C'' implies that $\alpha(C'') < s(C')/2c^4$. Therefore, $s(C') > 2c^3\alpha(C)$. Similarly, we can prove that $s(C) > 2c^3\alpha(C')$. □

We subdivide the small components into types or subsets $\mathcal{M}_1, \mathcal{M}_2, \ldots$. We say that a small component C is of *type i* and denote $C \in \mathcal{M}_i$ iff $c^{i-1} \leq \alpha(C) < c^i$.

Claim 8. *For each i, $|\mathcal{M}_i| \leq 4c^4$.*

Proof: Consider some $i \geq 1$, and assume that $|\mathcal{M}_i| > 4c^4$. Then in the optimal solution, there are more than $2c^4$ components of type i embedded on one of the sides of r. Denote these components by $C_1^i, C_2^i, \ldots, C_k^i$, $k > 2c^4$, and assume that vertices $r(C_j^i)$ are embedded in the optimal solution in this order, where $r(C_1^i)$ is embedded closest to r. It is easy to see that for any pair C, C' of small components, the distance between $r(C)$ and $r(C')$ is at least $\frac{\alpha(C)}{c}$. As the optimal embedding is non-contracting, for every $j = 1, \ldots, k-1$, there is a distance of at least $\alpha(C_j^i)/c \geq c^{i-2}$ between the embedding of $r(C_j^i)$ and $r(C_{j+1}^i)$. Therefore, $r(C_k^i)$ is embedded at a distance at least $kc^{i-2} > 2c^{i+2}$ from r. However, $d(r(C_k^i)) \leq \alpha(C_k^i) + c\alpha(C_k^i) \leq 2c^{i+1}$, and thus this distance is distorted by more than a factor of c in the optimal embedding. □

5.2 The Approximation Algorithm

Our algorithm consists of three major phases. In the first phase we compute the set $\mathcal{C}$ of components, after performing Procedure Partition. In the second phase, we solve the problem recursively for each one of the components $C \in \mathcal{C}$, where the threshold for the root condition becomes $H = D(r(C), r)$. In the final phase, we combine the recursive solutions to produce the final embedding.

Claim 9. *For each recursive call to our algorithm, there is an embedding of the corresponding instance with distortion c, that satisfies the root condition.*

Proof: Let C be a component, and let C' be a component obtained after decomposing C. We consider the recursive call in C'. Since C is just a subtree of T, it embeds into the line with distortion c. Let f be such an embedding of C with distortion c. W.l.o.g., we can assume that $r(C')$ is embedded to the left of $r(C)$. It suffices to show that f satisfies the root condition in component C'.

Observe that for the recursive call in C', the threshold value is $H = D(r(C), r(C'))$. All the edges of C' as not large w.r.to $r(C)$, thus all the vertices of C' are embedded to the left of $r(C)$. Assume now that the root condition is not satisfied for C'. This implies that there exists a component C'' that is obtained after decomposing C', such that $\alpha(C'') + s(C'') \geq cH$, and such that C'' is embedded to the left of $r(C')$. Thus, $f(r(C')) < f(l(C'')) < f(r(C))$. It follows that $|f(r(C')) - f(r(C))| > |f(r(C')) - f(l(C''))| \geq D(r(C'), l(C'')) = \alpha(C'') + s(C'') \geq cH = cD(r(C'), r(C))$, a contradiction. $\square$

The final embedding is produced as follows. First, partition the set $\mathcal{C}$ of components into two subsets $\mathcal{R}$, $\mathcal{L}$, containing the components to be embedded to the right and to the left of r, respectively. The partition procedure is explained below. The components in $\mathcal{L}$ are then embedded to the left of r, while the embedding of each component is determined by the recursive procedure call, and the embeddings of different components do not overlap. The order of components is determined as follows. For each small component C, let $f(C) = \alpha(C)$, and for each large component C', let $f(C') = s(C')/2c^4$. The order of embedding is according to $f(C)$, where the component C with smallest $f(C)$ is embedded closest to the root r. The embedding of components in $\mathcal{R}$ is performed similarly, except that the embedding of each component is the mirror image of the embedding returned by the recursive procedure call (so that the root condition holds in the right direction). We put enough empty space between the embeddings of different components to ensure that the embedding is non-contracting. In the rest of this section we show how to partition $\mathcal{C}$ into the subsets $\mathcal{R}$ and $\mathcal{L}$.

We start with large components. We translate the problem into an instance of 2SAT, as follows. We have one variable $x(C)$ for each large cluster C. Embedding C to the left of r is equivalent to setting $x(C) = T$. If two components C and C' are incompatible, we ensure that variables $x(C)$ and $x(C')$ get different assignments, by adding clauses $x(C) \vee x(C')$ and $\overline{x(C)} \vee \overline{x(C')}$. Additionally, if $s(C) + \alpha(C) > cH$, then we ensure that C is not embedded to the right of r by adding a clause $x(C) \vee x(C)$. The optimal solution induces a satisfying assignment to the resulting 2SAT formula, and hence we can find a satisfying assignment in polynomial time. The clusters C with $x(C) = T$ are added to $\mathcal{L}$ and all other clusters are added to $\mathcal{R}$.

Consider now any small cluster C. If $s(C) + \alpha(C) > cH$, then we add C to $\mathcal{L}$. Otherwise, if $s(C) + \alpha(C) \leq cH$, then there is at most one large component C' that has conflict with C. If such a component C' exists, then we embed C on the side opposite to that where C' is embedded. Otherwise, C is embedded to the left of r. Clearly, in any embedding consistent with the above decision the root condition is satisfied.

The analysis of this phase of the algorithm appears in the full version of this paper.

THEOREM 2. *The algorithm produces a non-contracting embedding with distortion bounded by* $c^{O(1)}$.

6. CONCLUSIONS AND OPEN PROBLEMS

In this paper we presented several results on approximation algorithms for minimizing distortion of embedding metrics into the line. This work naturally leads to several open problems. Perhaps the most glaring one is: does there exist a polynomial-time $c^{O(1)}$-approximation algorithm for embedding *general* metrics into the line (i.e., can the dependence on Δ in the approximation factor be completely removed) ? We conjecture that this is the case. Perhaps the first step towards designing such algorithm would be finding a simpler algorithm for the *tree-metric* case.

7. ADDITIONAL AUTHORS

8. REFERENCES

[1] R. Agarwala, V. Bafna, M. Farach-Colton, B. Narayanan, M. Paterson, and M. Thorup. On the approximability of numerical taxonomy: (fitting distances by tree metrics). *7th Symposium on Discrete Algorithms*, 1996.

[2] M Bădoiu. Approximation algorithm for embedding metrics into a two-dimensional space. *14th Annual ACM-SIAM Symposium on Discrete Algorithms*, 2003.

[3] M. Bădoiu, E. Demaine, M. Hajiaghai, and P. Indyk. Embeddings with extra information. *Proceedings of the ACM Symposium on Computational Geometry*, 2004.

[4] M. Bădoiu, K. Dhamdhere, A. Gupta, Y. Rabinovich, H. Raecke, R. Ravi, and A. Sidiropoulos. Approximation algorithms for low-distortion embeddings into low-dimensional spaces. *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms*, 2005.

[5] M. Bădoiu, P. Indyk, and A. Sidiropoulos. A constant-factor approximation algorithm for embedding unweighted graphs into trees. *AI Lab Technical Memo AIM-2004-015*, 2004.

[6] K. Dhamdhere. Approximating additive distortion of line embeddings. *Proceedings of RANDOM-APPROX*, 2004.

[7] K. Dhamdhere, A. Gupta, and R. Ravi. Approximating average distortion for embeddings into line. *Proceedings of the Symposium on Theoretical Aspects of Computer Science (STACS)*, 2004.

[8] J. Dunagan and S. Vempala. On euclidean embeddings and bandwidth minimization. *Proceedings of the 5th Workshop on Randomization and Approximation*, 2001.

[9] Y. Emek and D. Peleg. Approximating minimum max-stretch spanning trees on unweighted graphs. *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms*, 2004.

[10] M. Farach-Colton, S. Kannan, and T. Warnow. A robust model for finding optimal evolutionary tree. *Annual ACM Symposium on Theory of Computing*, 1993.

[11] U. Feige. Approximating the bandwidth via volume respecting embeddings. *Journal of Computer and System Sciences*, 60(3):510–539, 2000.

[12] J. Hastad, L. Ivansson, and J. Lagergren. Fitting points on the real line and its application to rh mapping. *Lecture Notes in Computer Science*, 1461:465–467, 1998.

[13] P. Indyk. Tutorial: Algorithmic applications of low-distortion geometric embeddings. *Annual Symposium on Foundations of Computer Science*, 2001.

[14] C. Kenyon, Y. Rabani, and A. Sinclair. Low distortion maps between point sets. *Annual ACM Symposium on Theory of Computing*, 2004.

[15] N. Linial. Finite metric spaces - combinatorics, geometry and algorithms. *Proceedings of the International Congress of Mathematicians III*, pages 573–586, 2002.

[16] N. Linial, E. London, and Y. Rabinovich. The geometry of graphs and some of its algorithmic applications. *Proceedings of 35th Annual IEEE Symposium on Foundations of Computer Science*, pages 577–591, 1994.

[17] J. Matoušek. Bi-lipschitz embeddings into low-dimensional euclidean spaces. *Comment. Math. Univ. Carolinae*, 31:589–600, 1990.

[18] MDS: Working Group on Algorithms for Multidimensional Scaling. Algorithms for multidimensional scaling. DIMACS Web Page.

[19] C. Papadimitriou and S. Safra. The complexity of low-distortion embeddings between point sets. *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms*, pages 112–118, 2005.

[20] J. B. Tenenbaum, V. de Silva, and J. C. Langford. A global geometric framework for nonlinear dimensionality reduction. `http://isomap.stanford.edu/`.

[21] W. Unger. The complexity of the approximation of the bandwidth problem. *Annual Symposium on Foundations of Computer Science*, 1998.

APPENDIX

A. GENERAL METRICS

Claim 2 For each i, with $1 \leq i \leq k$, $\max_{u \in V_i} D(u, v_i) \leq c^2W/2$.

Proof: Let $u \in V_i$. Consider the optimal embedding f. Since $f(v_1) = \min_{w \in X} f(w)$, and $f(v_k) = \max_{w \in X} f(w)$, it follows that there exists j, with $1 \leq j < k$, such that

$$\min\{f(v_j), f(v_{j+1})\} < f(u) < \max\{f(v_j), f(v_{j+1})\}.$$

Assume w.l.o.g., that $f(v_j) < f(u) < f(v_{j+1})$. We have $D(u, v_j) \geq D(u, v_i)$, since $u \in V_i$. Since f is non-contracting, we obtain $f(u) - f(v_j) \geq D(u, v_j) \geq D(u, v_i)$. Similarly, we have $f(v_{j+1}) - f(u) \geq D(u, v_i)$. Thus, $f(v_{j+1}) - f(v_j) \geq 2D(u, v_i)$. Since $\{v_j, v_{j+1}\} \in E(H')$, we have $D(v_j, v_{j+1}) \leq cW$. Thus, $c \geq \frac{f(v_{j+1})-f(v_j)}{D(v_{j+1},v_j)} \geq \frac{2D(u,v_i)}{cW}$. □

Claim 3 For each $r \geq 1$, and for each i, with $1 \leq i \leq k - r + 1$, $\sum_{j=i}^{i+r-1} |V_i| \leq c^2W(c + r - 1) + 1$.

Proof: Let $A = \bigcup_{j=1}^{i+r-1} V_i$. Let $x = \text{argmin}_{u \in A} f(u)$, and $y = \text{argmax}_{u \in A} f(u)$. Let also $x \in V_i$, and $y \in V_j$. Clearly, $|f(v_i) - f(v_j)| \leq cD(v_i, v_j) \leq cD_{H'}(v_i, v_j) \leq c^2W|i - j| \leq c^2W(r - 1)$. By Claim 2, we have $D(x, v_i) \leq c^2W/2$, and $D(y, v_j) \leq c^2W/2$. Thus, $|f(x) - f(v_i)| \leq cD(x, v_i) \leq c^3W/2$, and similarly $|f(y) - f(v_j)| \leq c^3W/2$. It follows that $|f(x)-f(y)| \leq |f(x)-f(v_i)|+|f(v_i)-f(v_j)|+|f(v_j)-f(y)| \leq c^3W + c^2W(r-1)$. Note that by the choice of x, y, and since the minimum distance in M is 1, and f is non-contracting, we have $\sum_{j=i}^{i+r-1} |V_i| \leq |f(x) - f(y)| + 1$, and the assertion follows. □

Claim 4 If $\{x, y\} \in E(H')$, with $x \in V_i$, and $y \in V_j$, then $D(v_i, v_j) \leq cW + c^2W$, and $|i - j| = O(c^2)$.

Proof: Since $\{x, y\} \in E(H')$, we have $D(x, y) \leq cW$. By Claim 2, we have $D(x, v_i) \leq c^2W/2$, and $D(y, v_j) \leq c^2W/2$. Thus, $D(v_i, v_j) \leq D(v_i, x)+D(x, y)+D(y, v_j) \leq cW+c^2W$. By Claim 1, we have $|i-j| = O(1+D(v_i, v_j)/W) = O(c^2)$. □