Dynamic Metric Embedding into ℓ_p Space

Kiarash Banihashem [1] MohammadTaghi Hajiaghayi [1] Dariusz R. Kowalski [2] Jan Olkowski [1] Max Springer [3]

Abstract

We give the first non-trivial decremental dynamic embedding of a weighted, undirected graph G into ℓ_p space. Given a weighted graph G undergoing a sequence of edge weight increases, the goal of this problem is to maintain a (randomized) mapping $\phi : (G, d) \to (X, \ell_p)$ from the set of vertices of the graph to the ℓ_p space such that for every pair of vertices u and v, the expected distance between $\phi(u)$ and $\phi(v)$ in the ℓ_p metric is within a small multiplicative factor, referred to as the *distortion*, of their distance in G. Our main result is a dynamic algorithm with expected distortion $O(\log^2 n)$ and total update time $O\left((m^{1+o(1)} \log^2 W + Q) \log(nW)\right)$, where W is the maximum weight of the edges, Q is the total number of updates and n, m denote the number of vertices and edges in G respectively. This is the first result of its kind, extending the seminal result of Bourgain (Bourgain, 1985) to the expanding field of dynamic algorithms. Moreover, we demonstrate that in the fully dynamic regime, where we tolerate edge insertions as well as deletions, no algorithm can explicitly maintain an embedding into ℓ_p space that has a low distortion with high probability.

1. Introduction

A low distortion embedding between two metric spaces, $M = (X, d)$ and $M' = (X', d')$, is a mapping f such that for every pair of points $x, y \in X$ we have

$$d(x, y) \leq d'(f(x), f(y)) \leq C \cdot d(x, y) \ ,$$

where C is often referred to as the *distortion* of such an embedding. Low-distortion embeddings have been extensively employed to simplify graph theoretic problems prevalent in the algorithm design literature (Indyk, 2001). This effectiveness stems primarily from the ability to represent any graph G using a metric, wherein distances correspond to the shortest paths between two nodes. However, computing numerous graph properties within such a metric is inherently challenging. Thus, by first embedding the graph into an "easy" metric, we can facilitate simplified problem-solving, albeit with an approximation factor determined by the distortion introduced by the embedding. For example, approximation algorithms for the sparsest cut (Linial et al., 1995), bandwidth (Blum et al., 1998) and buy-at-bulk (Awerbuch & Azar, 1997b) graph problems leverage embeddings into low-distortion metric spaces to obtain their near-optimal guarantees.

In the present work, we investigate fundamental embedding problems in the *dynamic* setting, where the input graph G is subject to modification at each iteration by an adversary. Specifically, we address the following question:

Problem 1. Is it possible to embed any graph G, undergoing a dynamic sequence of edge updates, into Euclidean (and more broadly the ℓ_p-metric) space with minimal distortion of the underlying metric's pairwise distances?

Unsurprisingly, the use of randomization is essential in demonstrating that such a data structure is indeed attainable. Most notably, we build upon the fundamental building blocks of Bourgain, Johnson and Lindenstrauss in further demonstrating the power of randomized decompositions of a graph to efficiently map such an input, undergoing dynamic updates, into Euclidean space for ease of computation with only polylogarithmic expected distortion of the original distances between nodes of G. These are the first results of their kind in the dynamic input setting.

1.1. Motivation

Metric Embedding. From a mathematical perspective, embeddings of finite metric spaces into normed spaces is a natural extension on the local theory of Banach spaces (J., 2002). The goal of this area of research is to devise mappings, f, that preserve pairwise distances up to an additive or multiplicative *distortion*. In tandem to ensuring this metric is not too heavily distorted, we also seek to ensure that the resulting embedding of a point in the original space has *low-dimension* (i.e. can be represented by small number of coordinates) to ensure the representation is spacially

[1]Department of Computer Science, University of Maryland, College Park, USA [2]School of Computer and Cyber Sciences, Augusta University, Georgia, USA [3]Department of Mathematics, University of Maryland, College Park, USA. Correspondence to: Max Springer <mss423@umd.edu>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024. Copyright 2024 by the author(s).

efficient.

Within this problem framework, the classic question is that of embedding metric spaces into *Hilbert space*. Considerable literature has investigated embeddings into ℓ_p normed spaces (see the survey (Abraham et al., 2006) for a comprehensive overview of the main results). Most crucially, the cornerstone of the field is the following theorem by Bourgain in 1985:

Theorem 1.1 ((Bourgain, 1985)). *For every n-point metric space, there exists an embedding into Euclidean space with distortion $O(\log n)$.*

This landmark result is foundational in the theory of embedding into finite metric spaces. Moreover, it was further shown in (Linial et al., 1995) that Bourgain's embedding yields an embebdding into *any* ℓ_p-metric with distorition $O(\log n)$ and dimension $O(\log^2 n)$ – demonstrating a highly efficient and precise algorithm.

We highlight that the above results are of immense value to the field of computer science in the age of big data where the construction of appropriately sized data structures is no longer efficient (or even feasible). For instance, in the field of social media analysis, processing billions of daily tweets to identify trending topics and sentiment analysis would require impractical amounts of storage and computational resources without dimension reduction techniques like topic modeling algorithms (Church, 2017; Subercaze et al., 2015). It is thus essential to reduce these inputs to a more approachable metric space to prevent computational bottle-necking. In this paper, we present the first extension on these seminal tools to the emerging domain of *dynamic algorithms*. Specifically, we maintain a polylogarithmic distortion embedding into the ℓ_p-metric through a sequence of updates to the input graph.

Dynamic Algorithm. A dynamic graph algorithm is a data structure that supports edge insertions, edge deletions, and can answer queries on certain properties of the input with respect to the original space's metrics. While trivially one can run a static algorithm on the graph after each update and rebuild a structure equipped to answer queries, the now large body of work on dynamic algorithms works to devise solutions with considerably faster update and query times. In the present work, we maintain a dynamic data structure that both reduces the dimension of the input for ease of computation and exhibits only a modest expansion of the original metric's pairwise distances in expectation.

Similar to the fundamental motivation underlying metric embeddings, the emergence of big data has intensified the need for dynamic algorithms capable of efficiently storing representations of massive input graphs, while promptly adapting to any changes that may occur on a variety of machine learning and optimization problems (Bhattacharya et al., 2022; Dütting et al., 2023). As an illustrative example, consider the problem of maintaining connectivity information in a large graph that undergoes edge insertions and deletions – an essential problem in the field of route planning and navigation. In a static scenario, the solution can be trivially achieved by rebuilding the shortest paths between nodes using Djikstra's algorithm on every source vertex after each update to the graph. However, it is easy to see that for connectivity graphs employed in big data systems, this procedure quickly becomes intractable. Recent advancements in the field of dynamic algorithms have revealed that it is possible to maintain such connectivity information with considerably less overhead in terms of the update time to the data structure without a large loss of accuracy for the paths (Bernstein, 2009; Roditty & Zwick, 2004; 2012). This capacity to adapt data structures to effectively handle diverse queries is rapidly transitioning from being merely helpful to absolutely essential. Building upon this existing body of literature, we present a novel contribution by developing a dynamic embedding structure tailored to capturing the distances between nodes in a graph, specifically within the context of the simplified ℓ_p metric – a highly useful computation in the field of dimension reduction for big data. Importantly, our approach guarantees a polylogarithmic update time, thereby striking a balance between efficiency and accuracy.

1.2. Our Results

We first explore the *decremental* setting, where edge weights can only increase dynamically (i.e., nodes move further apart); this is the setting under which our primary algorithmic contributions are effective. For the *fully* dynamic setting which allows both increases and decreases in edge weights, we show a partial negative result proving that maintaining an embedding into the ℓ_p-metric explicitly that has low distortion with high probability is not feasible. Here *explicitly* maintaining an embedding means that the entire embedding is updated efficiently, rather just reporting any changes to the data structure (see Section 2 for a more precise definition of these problem regimes).

Theorem 1.2. *There is no fully dynamic algorithm that can explicitly maintain a dynamic embedding into ℓ_p space with high probability.*

Though computation is efficient in the target space, we demonstrate that an adversarially selected sequence of updates to the graph can force an update of the embedding for $\Omega(n)$ nodes in each step which becomes intractable to maintain. Intuitively, this result is derived from the fact that ℓ_p space is *complete* and *locally compact*, while an input graph can become essentially disconnected through edge weight updates resulting in an infeasible embedding. We expand more formally on this result in Section 3.

The main idea underpinning our primary algorithmic result is a novel combination of the static randomized decomposition of a graph (as utilized by Bourgain) with a decremental clustering algorithm to maintain an embedding into ℓ_p space that exhibits $O(\log^2 n)$ stretch and can answer distance queries with polylogarithmic update time. Our algorithmic result is stated formally as follows.

Theorem 1.3. *For every graph G with max edge weight W and a metric ℓ_p, there is a decremental dynamic algorithm that maintains an embedding, $\rho : V \to \mathbb{R}^{\log(nW)}$, for the metric induced by the dynamically updated graph G into ℓ_p space of dimension $\log(nW)$ that has expected (over the internal randomization of the algorithm) stretch at most $O(\log^2 n)$ and its running time is at most $O\left((m^{1+o(1)} \log^2 W + Q) \log(nW)\right)$ with high probability[1], where Q denotes the total number of updates. Within this running time, the algorithm explicitly outputs all changes to the embedding and can answer distance queries between pair of vertices in $O(\log(nW))$ time.*

To prove the guarantees of this algorithm, we require an alternative, constructive proof of Bourgain's lemma. Our algorithm is different from standard approaches to the problem which can be classified as "Frechet embeddings." In these embeddings, each coordinates $\rho_i(v)$ takes the form of $d_G(v, S_i)$ where S_i is a specific set. However, these approaches are not suitable for the dynamic setting due to limitations in analyzing their upper bound on $\|\rho(u) - \rho(v)\|_p$ for every given u and v. Specifically, the distances can be maintained only approximately at best, prohibiting us from obtaining an upper bound.

Starting from the static case, we introduce the notion of a (random) (β, R, ϵ)-distance preserving cut. There are two main properties of a (β, R, ϵ)-distance preserving cut. Ignoring for now the technical ϵ parameter of this notation, the parameters β and R control the following. First, we require that the probability that two vertices are in different sets is at most β times the distance between these vertices in G. Intuitively, we can expect many close vertices to be on the same side of the cut. On the other hand, for every pair of vertices whose distance in G is larger than R, we require probability at least $\frac{1}{2}$ that they are on different sides of the cut. The rationale behind the latter property is that such a cut will, with constant probability, properly distribute vertices that are of distance at least R in G. We then construct $O(\log(nW))$ such cuts, where the i-th cut corresponds to a different choice of the distance steering parameter R, i.e. $R_i = 2^i$. The final embedding is made by assigning every vertex a vector of $\log nW$ coordinates, one coordinate for corresponding to each parameter choice R_i. For every cut we denote its two sides as "left" and "right". If a vertex is on the left side of the i-th cut, we set its i-th coordinate to 0; if it is on the right side, we set the coordinate to R_i. Using both aforementioned properties of a (β, R, ϵ)-distance preserving cut, we show that such an assignment is an embedding with $O(\log^2 n)$ stretch.

To implement this algorithm in the dynamically changing graph G, we prove that (β, R, ϵ)-distance preserving cuts can be efficiently obtained from a (β, δ)-weak decomposition of G, a probabilistic graph partitioning introduced by Bartal (Bartal, 1996). In this decomposition, we partition vertices of G into clusters such that the distance (with respect to G) between every pair of vertices in a cluster is at most δ, but on the other hand, for every edge the probability that this edge connects two *different* clusters is at most β times its weight. To proceed to (β, R, ϵ)-distance preserving cuts, we augment this construction by randomly assigning each cluster to one of the two sides of the cut. In the analysis, we manage to show that such simple random experiments guarantee the properties we require from a (β, R, ϵ)-distance preserving cut. On the other hand, provided that we are able to dynamically maintain (β, δ)-weak decomposition of G, it is simple to update the random assignment after each update. To deal with a (β, δ)-weak decomposition of G under dynamic updates, we lean on the result of (Forster et al., 2021) who showed how to maintain such a decomposition under *edge deletions*. We observe that their framework, with few technical changes, translates to our settings.

We discuss the details of the underlying *static* tools used to maintain this structure in Section 4 and proceed to augment these procedures to maintain edge weight updates in Section 5. Moreover, we note that the embedding can be used to implement a dynamic distance oracle (all-pairs shortest paths), as for each two vertices in the graph, we can estimate their distances efficiently by calculating the distance between their embeddings. While our distance guarantees only hold in expectation, the update time of a distance oracle based on our algorithm nearly matches the best known bounds for the APSP problem for $O(\log^2 n)$ stretch (Chechik, 2018; Forster et al., 2023), which further shows the tightness of our analysis.

1.3. Related Work

Metric Embedding. The foundational result for the algorithmic applications of metric embedding is that of Bourgain in 1985 (Bourgain, 1985) which embeds into any ℓ_p metric with logarithmic distortion. When the input metric is already the ℓ_2 metric, the result of Johnson and Lindenstrauss (Johnson et al., 1986) shows that its size can be reduced to $O(\log n/\varepsilon^2)$ with $(1+\varepsilon)$ distortion for $\varepsilon > 0$. Recent works have studied lower bounds for the minimum number of dimensions necessary for this compression; e.g., see (Larsen

[1]Throughout the paper, we say that an event holds with high probability (whp for short), if its probability is at least $1 - n^{-a}$ for some absolute constant a.

& Nelson, 2017). To the best of our knowledge, these embedding results have no analogous algorithm in the dynamic setting, which we formulate in the present work.

While ℓ_p space is extremely useful for functional approximation and other challenging mathematical problems, there also exists a line of research on the embeddings of an input metric to a *tree metric* which inherently lends itself to dynamic problems. For embedding into these tree structures, an emphasis is placed on algorithms for *probabilistic tree embeddings* (PTE) where the host metric is embedded into a *distribution* of trees. Concretely, given a graph G, the objective is to find a distribution over a set τ of trees such that distances in G do not get contracted and the expected distances over the randomly sampled tree distribution do not exceed a multiplicative *stretch* of α (stretch here can be considered interchangeable with the concept of distortion). The preliminary work on such embeddings from Bartal (Bartal, 1996) demonstrated that by a "ball growing" approach, we can embed any graph with $O(\log^2 n)$ stretch with a nearly equivalent lower bound of $\Omega(\log n)$ stretch for any such embedding. This work was later improved to obtain a PTE procedure with optimal $O(\log n)$ stretch (Fakcharoenphol et al., 2003) which has applications in problems for metric labeling (Kleinberg & Tardos, 2002), buy-at-bulk network design (Awerbuch & Azar, 1997a), vehicle routing (Charikar et al., 1998), and many other such contexts (Bartal, 2004; Garg et al., 2000). Our dynamic emebdding procedure combines this ball growing approach with a decremental clustering procedure to efficiently maintain an embedding into the ℓ_p-metric.

Dynamic Embedding. Closely related to our work is the study of dynamic embeddings into trees. The work of (Forster & Goranci, 2019) initiates the study on the dynamic maintenance of low-stretch such *spanning* trees, devising an algorithm that yields an *average distortion* of $n^{o(1)}$ in expectation with $n^{1/2+o(1)}$ update time per operation. This result was later improved to $n^{o(1)}$ average distortion and update time bounded by $n^{o(1)}$ (Chechik & Zhang, 2020).

The restriction of these prior works to the maintenance of spannning trees is an inherently more difficult and limited problem instance. To improve upon the above bounds, (Forster et al., 2021) removes this restriction and designs an embedding procedure that guarantees an *expected* distortion of $n^{o(1)}$ in $n^{o(1)}$ update time, or $O(\log^4 n)$ stretch with $m^{1/2+o(1)}$ update time when embedding into a distribution of trees. This work also devises a decremental clustering procedure that we build upon in the present work to devise our embeddings. We additionally note that the expected distortion objective more closely aligns with our primary result, however our embedding into the ℓ_p -metric is better suited for the class of NP-hard optimization problems whose approximation algorithms rely on the geometry of Euclidean space such as sparsest cut (Arora et al., 2005; Aumann & Rabani, 1998; Chawla et al., 2008), graph decompositions (Arora et al., 2009; Linial et al., 1995), and the bandwidth problem (Dunagan & Vempala, 2001; Feige, 1998; Krauthgamer et al., 2004). Moreover, our guarantees remove the dependence on the input weight parameter W, yielding a distortion that is polylogarithmic in n alone.

Similar to the present work is the study of *dynamic distance oracles* as originally studied by (Thorup & Zwick, 2005) in the static setting, and later extended to the decremental setting with a data structure which maintains the distance between any two points from the input metric with $O(2k-1)$ stretch, $\tilde{O}(mn)$ total update time and $O(m+n^{1+1/k})$ space (where k is any positive integer) (Roditty & Zwick, 2012). This result can be further improved to a distortion of $1+\varepsilon$ with $\tilde{O}(n^2)$ space for every $\varepsilon > 0$. (Chechik, 2018) further present a decremental algorithm for the all pairs shortest path (APSP) problem which admits $(2+\varepsilon)k-1$ distortion with total update time of $O(mn^{1/k+o(1)}\log(nW))$ and query time $O(\log\log(nW))$. Our embedding which generalizes this notion of distance oracle yields a nearly equivalent update time for $O(\log^2 n)$ stretch, further demonstrating the tightness of our analysis.

In the next section, we precisely define the mathematical framework and formalization within which our algorithmic techniques reside.

2. Model and Preliminaries

Let $G = (V, E)$ be a weighted, undirected graph on n vertices with (at most) m edges of positive integer weights in the range from 1 to W, where W is a fixed parameter known to the algorithm. For an edge $(u, v) \in E$, we denote its weight by $w_G(u, v)$. For every pair of nodes $u, v \in V$, let $d_G(u, v)$ be the length of the shortest weighted path between nodes u, v in G, where we define the weight of a path as the sum of the weights of its edges. Throughout, we let Δ denote the max distance between any two nodes (note that $\Delta \le nW$). We note that (V, d_G) is a metric space.

Given a set of vertices $V' \subseteq V$, we define the *weak diameter* of V' as the maximum distance between the vertices of V' in the original graph, i.e., $\text{wdiam}(V') = \sup_{u,v\in V'} d_G(u, v)$. For all $u \in V$ and $r \ge 0$, let $B_G(u, r)$ denote the set of all vertices that are within distance r from u in the graph G, i.e., $B_G(u, r) := \{\, v \in V : d_G(u, v) \le r \,\}$.

Metric Embedding. The objective of this paper is to construct and maintain an embedding of the metric defined by an input graph G to an ℓ_p metric space without distorting the original distances by too much. More formally, given a metric space (X, d_X), an injective mapping $f : G \to X$

is called an *embedding*, from G into X. We define the *expansion* (or stretch) and the *contraction* of the embedding f, respectively, as:

$$\text{expans}(f) = \sup_{u,v\in V; u\neq v} \frac{d_X(f(u), f(v))}{d_G(u,v)}$$

$$\text{contr}(f) = \sup_{u,v\in V; u\neq v} \frac{d_G(u,v)}{d_X(f(u), f(v))} .$$

We define the distortion of the embedding f as distort$(f) =$ expans$(f) \cdot$ contr(f). Note that any embedding f satisfies $\frac{1}{\text{contr}(f)} \cdot d_G(u,v) \leq d_X(f(u), f(v)) \leq \text{expans}(f) \cdot d_G(u,v)$. The embeddings in this paper are random functions, and are constructed by randomized algorithms. Given a random embedding $f : V \to X$, we define its *expected distortion* as the smallest value $\alpha > 0$ for which there exist positive values a, b satisfying $ab = \alpha$ such that for all $u, v \in V$: [2]

$$\frac{1}{a} \cdot d_G(u,v) \leq \mathbb{E}\left[d_X(f(u), f(v))\right] \leq b \cdot d_G(u,v) . \quad (1)$$

In this paper, we focus on embeddings into the ℓ_p metric space. In this metric space, the ground set X equals $\mathbb{R}^d$, for some positive integer d, and for every pair of points $x, y \in X$, the distance d_X is defined as

$$d_X(x,y) = \|x - y\|_p = \left(\sum_{i=1}^{d} |x_i - y_i|^p\right)^{1/p} ,$$

where x_i and y_i refer to the i-th coordinate of x and y, respectively.

Dynamic Model. We consider a model where the underlying input graph G undergoes a sequence of updates as specified by an *oblivious* adversary. We assume that the adversary knows the algorithm, but does not have access to the random bits the algorithm uses. We use $G_0, G_1, \ldots$ to denote the corresponding sequence of graphs, where G_i refers to the graph after i updates. Throughout, we will use Q to denote the total number of updates to an input graph. This sequence is fixed by the adversary before the execution of the algorithm, but is revealed to the algorithm gradually, one by one, in online manner. Our goal is to explicitly maintain an embedding after each update, as formally defined below:

Definition 2.1 (Maintain)**.** We say that a dynamic algorithm $\mathcal{A}$ **explicitly maintains** an embedding of the input graph into a metric space (X, d_X) if there exists a sequence of mappings $\phi_0, \phi_1, \ldots$ where $\phi_i : V \to X$ and $\mathcal{A}$ outputs the changes in ϕ after every update. Formally, after the update t, the algorithm should output v and $\phi_t(v)$ for all v such that $\phi_t(v) \neq \phi_{t-1}(v)$.

[2]Throughout the paper, we mostly consider $a = 1$. As such, we sometimes use distortion and stretch interchangeably since we are only concerned with the expansion of distances between points.

We operate in the *decremental* setting and assume that each update takes the form of an edge weight increase, i.e., for an edge $(u, v) \in E$, the value of $w_G(u, v)$ increases. We note that this is slightly different from the standard definition of the decremental setting which permits the *deletion* of edges in the input graph. The deletion of an edge can lead the input graph to potentially become disconnected, which means we may have $d_{G_t}(u,v) = \infty$ for some time step t and $u, v \in V$. This is problematic, however, because regardless of the value of $\phi_t(u)$ and $\phi_t(v)$, we will always have $\|\phi_t(u) - \phi_t(v)\|_p < \infty$ because the ℓ_p metrics do not allow for infinite distances.[3] This in turn means that we cannot satisfy the bounds for expected distortion (Equation (1)), and as such cannot design a low-distortion embedding. To avoid this issue, we restrict the updates to edge weight increases only, and we note that in practice the removal an edge can be simulated by choosing a large W as the dependence of our bounds on W will be polylogarithmic. Thus, edge weight increases serve as a necessary stand-in for edge deletions as both will lead to pairwise distances increasing.

In the section that follows, we will show that maintaining a fully dynamic embedding, where edge weights are subject to both increases and decreases, that has low distortion with high probability is unfeasible in the ℓ_p-metric space if the distortion bounds hold. This limitation underpins the rationale for the above decremental problem setting we introduce.

3. Lower Bound for Explicit Maintenance of Fully Dynamic Embeddings

We first present an (oblivious) adversarial construction of edge weight modifications to a graph in the *fully dynamic* model that cannot be *explicitly* maintained in the geometry of ℓ_p space without needing to modify the embedding for every node in the original graph. We highlight that this is a high probability result whereas the main algorithmic results we obtain hold in expectation.

Theorem 3.1. *Any fully dynamic algorithm that maintains an embedding into the ℓ_p-metric space which guarantees a distortion of at most $o(W)$ with high probability must have an update time at least $\Omega(n)$.*

Proof. Let $\mathcal{A}$ be a fully dynamic algorithm which guarantees a stretch of at most W with high probability. Consider an input graph G that consists of two separate complete graphs on n vertices, H and H', comprised of unit edges. Further consider two fixed vertices $v \in H, v' \in H'$. If there

[3]Note that our algorithm can be run in parallel on multiple connected components after becoming disconnected. However, we here restrict our attention to the guarantees on a single connected component.

is a unit edge between these two vertices, then the distance of all elements in H and H' is at most 3 in the graph metric, and therefore in the ℓ_p embedding cannot be more than $O(W)$.

Now, assume an adversary increases the edge weight connecting the vertices v and v' to a value of W. In the original graph metric, all pairwise distances between the nodes of H and H' must now be at least W. Therefore, the embedded points of one cluster (H or H') must be updated so as to not contract the original metric and maintain the distortion of at most W with high probability (see Figure 1 for a depiction of this construction). Therefore, the algorithm $\mathcal{A}$ must update the embedding for all n nodes of one of the complete components of G to be at least W away from the other with respect to the ℓ_p norm and satisfy the distortion constraints with high probability. Thus, we charge at least $\Omega(n)$ to the update time of $\mathcal{A}$ in the worst case for the maintenance of the embedding. [4] Moreover, we cannot amortize this worst case update occurrence since, in the subsequent iteration, the adversary can change the edge weight back 1 and repeat the cycle – resulting in $\Omega(n)$ updates per iteration. $\square$

Though this sequence is simplistic, it highlights the inherent limitations of embedding into a complete and locally compact metric like the ℓ_p normed space. We additionally remark that, in the expected distortion setting of our algorithmic result, this lower bound does not persist since a large increase in the expected pairwise distances between nodes does not necessarily imply the embedding for every pair of points has been updated.

4. Static algorithm

We proceed to present our algorithm by first presenting the static partitioning procedure, which is used to initialize our data structure and is subsequently maintained through the sequence of updates specified by the adversary. While our ideas are based on prior work, to our knowledge, this static algorithm is a novel construction that has not appeared before in the literature.

[4]Formally, for each pair of vertices in $H \times H'$, at least one of them needs to be updated. Since there are $\Omega(n^2)$ pairs and each vertex update resolves the issue for $O(n)$ pairs, we need $\Omega(n)$ vertex updates.

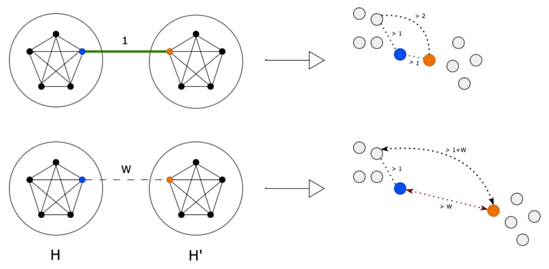

Figure 1. Adversarial sequence of graph updates

4.1. Distance Preserving Cuts

Our algorithm dynamically maintains an embedding based on a set of cuts in the graph, where each cut is designed to separate vertices with distances above some threshold R, while simultaneously preserving distances of the vertices in the graph. We formally define the notion of a *distance preserving cut*.

Definition 4.1 (Distance preserving cut)**.** Given a graph $G = (V, E)$, let $S \subseteq V$ be a random subset of the vertices. For vertices u, v, let $\texttt{cut}_{u,v}$ denote the event that u, v are on different sides of the partition $(S, V \backslash S)$, i.e.,

$$\texttt{cut}_{u,v} = \{\, u \in S \text{ and } v \notin S \,\} \text{ or } \{\, u \notin S \text{ and } v \in S \,\} \ .$$

We say that S is a (β, R, ϵ)-distance preserving cut, or (β, R)-cut for short, if it has the following three properties:

- $\Pr[\texttt{cut}_{u,v}] \leq \beta \cdot d(u,v)$ for every u, v and
- $\Pr[\texttt{cut}_{u,v}] = 0$ for every u, v such that $d(u,v) < \epsilon$,
- $\Pr[\texttt{cut}_{u,v}] \geq \frac{1}{2}$ for every u, v such that $d(u,v) > R$.

Following in the algorithmic technique of decomposing the graph into these smaller sets with desirable pairwise distance bounds is a refinement on the ball-growing approach of Bartal (Bartal, 1996) and the padded decomposition at the heart of Bourgain's embedding results (Bourgain, 1985). Most importantly, this efficient cut set construction allows us to contract edges which are small enough to ignore in the approximation factor and also provide logarithmic distortion bounds on the larger paths – a fact that will be verified in the following analysis.

The main result in this section is the following lemma that guarantees the existence of such cut sets and will be used heavily in our pre-processing graph decomposition at various granularities which in turn leads to the desired distortion bounds promised in Theorem 1.3.

Lemma 4.2. *For every $1 \leq R \leq \Delta$, there exists a (β, R, ϵ)-distance preserving cut with $\beta = O\left(\frac{\log n}{R}\right)$ and $\epsilon = O\left(\frac{R}{n}\right)$.*

We now present the proof of this lemma which uses the randomized decomposition method of Bartal (Bartal, 1996) in conjunction with a novel probabilistic compression procedure. First, we review the definition of an R-partition of a graph G (as originally defined in Bartal (Bartal, 1996)).

Definition 4.3. An R-partition of G is a collection of subsets of vertices $P = \{V_1, ..., V_k\}$ such that

- For all $i \in [k]$, $V_i \subseteq V$ and $\bigcup_{i \in [k]} V_i = V$.

Algorithm 1 Low-Diameter Randomized Decomposition (LDRD) (Bartal, 1996)

```
1: Input: Graph $G = (V, E)$ and parameter $1 \leq R \leq \Delta$
2: Output: An LDRD of G, denoted $\mathcal{C} = \{C_i\}_{i=1}^n$
3: Contract edges of $G$ which are at most $\frac{R}{2n}$
4: Set $U \leftarrow V$
5: for $u \in U$ do
6:    Sample $r \sim G(\beta)$
7:    $C(u) \leftarrow u$
8:    $C(u) \leftarrow C(u) \cup \{v \in U : d_G(u, v) \leq r\}$
9:    $U \leftarrow U \setminus C(u)$
10: end for
11: Return: $\{C_i\}_{i=1}^n$
```

- For all $i, j \in [k]$ such that $i \neq j$, $V_i \cap V_j = \emptyset$.
- Let $C(V_i)$ denote the subgraph of G induced on the vertices of V_i. Such a subgraph is referred to as a "cluster" of the partition and for every $i \in [k]$, the weak diameter of V_i is bounded above as $\text{wdiam}(C_i) \leq R$.

Next, we give the definition of a (β, R)-weak decomposition - a modificiation on the R partition that probabilistically ensures vertices close to each other appear in the same cluster.

Definition 4.4. Given a graph $G = (V, E)$, let $\mathcal{C} = \{C_1, \dots C_k\}$ be a (random) partitioning of the vertices. For every $u \in V$, let $C(u) \in [k]$ denote the index i such that $u \in C_i$. We say that $\mathcal{C}$ is a (β, R)-weak decomposition of G if for every u, v we have

$$\Pr[C(u) \neq C(v)] \leq \beta \cdot d(u, v)$$

and for every i and pair $u, v \in C_i$ we have $d(u, v) \leq R$.

Following Bartal (Bartal, 1996), we prove that for all R and $\beta \geq \frac{8 \log n}{R}$ there exists a (β, R)-weak decomposition of G that has the additional property that vertices which are closer than $\frac{R}{2n}$ are necessarily in the same cluster. Formally,

Theorem 4.5. *Given a graph $G = (V, E)$ with parameters $1 \leq R \leq \Delta$ and $\beta \geq \frac{8 \log n}{R}$, there exists a (β, R)-weak decomposition $\{C_1, ..., C_k\}$ such that for every pair of vertices $u, v \in V$:*

- $\Pr[C(u) \neq C(v)] \leq \frac{8 \log(n)}{R} \cdot d_G(u, v)$
- *If $d_G(u, v) < \frac{R}{2n}$ then u and v are in the same cluster.*

Given this randomized decomposition of our graph, we can construct the desired "cut" set that preserves distances by a simple random compression scheme that combines clusters from the above process. Specifically, we take each cluster from Theorem 4.5 and independently assign to one side of the constructed cut, grouping all the clusters into one

Algorithm 2 Randomized (β, R, ϵ)-Cut Decomposition

```
1: $\mathcal{C} \leftarrow$ LDRD$(G, R)$
2: $S \leftarrow \emptyset$
3: for $C \in \mathcal{C}$ do
4:    Pick $\tau \in \{0, 1\}$ uniformly at random
5:    if $\tau = 1$ then
6:       $S \leftarrow S \cup C$
7:    end if
8: end for
9: Return: $(S, V \setminus S)$
```

of two groups. Within these groups we then merge the clusters to obtain our desired cut sets, S and $V \setminus S$. The following lemma verifies that this is a distance preserving cut and the pseudocode is presented in Algorithm 2 for clarity. The proof is deferred to the appendix due to space constraints.

Lemma 4.6. *Given a value $1 \leq R \leq \Delta$, let $\{C_i\}_{i=1}^k$ be the weak decomposition of the graph satisfying the properties of Theorem 4.5, and define the cut S as $S := \cup_{i \in [k]: x_i = 1} C_i$, where $x_1, \dots, x_k$ is a sequence of i.i.d Bernoulli variables with parameter $\frac{1}{2}$. The cut S is a (β, R, ϵ)-distance preserving cut with $\beta := O(\frac{\log(n)}{R})$ and $\epsilon = O(\frac{R}{n})$.*

4.2. Embedding Procedure

We now proceed to show how to obtain an embedding of the graph using the distance preserving cuts of the previous section. Let Δ be an upper bound on the diameter of the graph G. We define our embedding that builds upon Definition 4.1 as follows.

Definition 4.7. Given a sequence of cuts $(S_1, \dots, S_r)$ and parameters $(R_1, \dots, R_r)$, we define the characteristic embedding of $(S_i, R_i)_{i=1}^r$ as a mapping $\rho : V \to \mathbb{R}^r$ that sets the i-th coordinate of $\rho(v)$ to R_i if $v \in S_i$ and to 0 otherwise, i.e., $\rho(v)_i := R_i \cdot \mathbb{1}\{v \in S_i\}$.

We note the difference our embedding procedure and the existing embedding procedures into ℓ_p space. The standard approach to the problem is to use *Frechet embeddings*; each coordinate $\rho_i(v)$ is of the form $d_G(v, S_i)$ for some set S_i. These sets are either obtained randomly, or using the partitioning scheme of the Fakcharoenphol-Rao-Talwar (FRT) embedding (Fakcharoenphol et al., 2003). These procedures are not well-suited for the dynamic setting, however because of the analysis of their upper bound on $\|\rho(u) - \rho(v)\|_p$ for every pair u, v. Specifically, in order to bound $\|\rho(u) - \rho(v)\|_p$, the approaches rely on

$$|\rho_i(u) - \rho_i(v)| = |d_G(u, S_i) - d_G(v, S_i)| \leq d_G(u, v),$$

where the inequality follows from the triangle inequality. In the dynamic setting however, (efficiently) maintaining distances can only be done approximately. This means that $\rho_i(u)$ and $\rho_i(v)$ would each be within a $(1+\epsilon)$ factor of $d_G(u, S_i)$ and $d_G(v, S_i)$ which would result in a degradation of our guarantees when maintained dynamically.

We now leverage the key characteristics for a set of distance preserving cuts to demonstrate that the corresponding characteristic embedding preserves the original distances in the ℓ_p-metric with only polylogarithmic distortion.

Theorem 4.8. *Given a graph $G = (V, E)$ and a parameter Δ such that $\Delta \geq diam(G)$, let $S_1, \dots, S_{\log(\Delta)}$ be (random) subsets of V such that S_i is a $(\beta_i, R_i, \epsilon_i)$-cut with $R_i = 2^{i+1}$ and $(\beta_i, \epsilon_i) = (O(\frac{\log n}{R}), O(\frac{R}{n}))$, and let $\rho : V \to \mathbb{R}^{\log \Delta}$ be the characteristic embedding of these cuts. For every pair of vertices u, v and any $p \in [1, \infty)$:*

$$\frac{1}{4} \cdot d(u, v) \leq \mathbb{E}\left[\|\rho(u) - \rho(v)\|_p\right] \leq O(\log^2 n) d(u, v).$$

Equipped with this static embedding that only distorts pairwise distances by a polylogarithmic factor, we proceed to adapt the structure to efficiently modify the cut-set decomposition of G through a sequence of (adversarially chosen) edge weight increases.

5. Dynamic Algorithm

In this section, we prove Theorem 1.3[5]. We do it by constructing an algorithm that dynamically maintains an embedding of a metric induced by a graph G into ℓ_p space of dimension $O(\log \Delta)$.

Our construction starts by observing a reduction. Informally, in the next theorem we show that in order to maintain dynamically the desired embedding, it is enough to have a dynamic algorithm that for every $1 \leq R \leq 2\Delta$ maintains a $(\frac{\log n}{R}, R, \frac{R}{n})$-distance preserving cut.

Theorem 5.1. *Assume we are given an algorithm $\mathcal{A}$ that takes as input the parameters R and ϵ which decrementally maintains a (β, R, ϵ) distance preserving cut S for a graph G undergoing edge weight increases, outputting changes to S after each such update, where $\beta := \frac{\log n}{R}$ and $\epsilon = \frac{R}{n}$. Assume further that the total running time of the algorithm $\mathcal{A}$ is bounded by $t(m, n)$ whp. Then there is a decremental dynamic algorithm that maintains an embedding of the vertices $\rho : V \to \mathbb{R}^{\log \Delta}$ that has expected (over the internal randomization of the algorithm) stretch at most $O(\log^2 n)$ and running time at most $O\left(t(m, n) \log \Delta\right)$, whp.*

We now show how to maintain a distance preserving cut dynamically, which in turn leads to a dynamic embedding algorithm via Theorem 5.1 and completes the proof of the main result, Theorem 1.3. We start by observing that a (β, δ)-weak decomposition of G can be dynamically maintained. We here highlight that the authors of (Forster et al., 2021), building upon the approach of (Chechik & Zhang, 2020), have already proved that a (β, δ)-weak decomposition of G can be dynamically maintained under *edge deletions* (Corollary 3.8). The proof of our observation involves adapting their techniques to the slightly modified definition of dynamic changes we invoke here to handle the continuous nature of ℓ_p space.

[5]Because of the page limit we avoid repeating statements of longer theorems.

Lemma 5.2. *For every $\beta \in (0, 1)$ and $\delta = (6(a+2)(2 + \log m) \ln n)\beta^{-1} = O(a\beta^{-1} \log^2 n)$, where $a \geq 1$ is a given constant controlling the success probability, there is a decremental algorithm to maintain a probabilistic weak (β, δ)-decomposition of a weighted, undirected graph undergoing increases of edge weights that with high probability has total update time $O(m^{1+o(1)} \log^2 W + Q)$, where Q is the total number of updates to the input graph, and (within this running time) is able to report all nodes and incident edges of every cluster that is formed. Over the course of the algorithm, each change to the partitioning of the nodes into clusters happens by splitting an existing cluster into two or several clusters and each node changes its cluster at most $O(\log n)$ times.*

Equipped with this tool, we can present the main contribution of this section - the maintenance of a (β, R, ϵ)-distance preserving cut under dynamic edge weights increases.

Lemma 5.3. *For every $0 \leq R \leq 2\Delta$, there is a decremental dynamic algorithm that maintains a $\left(\frac{\log n}{R}, R, \epsilon\right)$-distance preserving cut a of weighted, undirected graph G. Its total update time is $O(m^{1+o(1)} \log^2 W + Q)$ with high probability, where Q is the total number of updates to the input graph, and, within this running time, explicitly reports all changes to the maintained cut.*

The synthesis of these two lemmas with the result of Theorem 5.1 yields the overall dynamic embedding of Theorem 1.3.

6. Conclusion

We here present the first dynamic embedding into ℓ_p space which is equipped to handle edge weight increases – a non-trivial extension of the seminal Bourgain and JL embedding results (Bourgain, 1985; Johnson et al., 1986). Most notably, our embeddings produce only a polylogarithmic distortion of the base metric and exhibit an update time on par with the best known results for the APSP and other embedding based problems. Our embedding procedure additionally reports any modifications within polylogarithmic time and is naturally well suited to the class of NP-hard optimization problems which rely on Euclidean geometry for approximations to the optimal solution. To supplement our algorithmic result, we further present a lower bound for the fully dynamic setting where edge weights can be increased or decreased. In particular, we show that no algorithm can achieve a distortion better than $o(W)$ with high probability without inheriting an update time of $\Omega(n)$ which makes the procedure inefficient in practice.

Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none which we feel must be specifically highlighted here.

Acknowledgements

We thank the anonymous reviewers for their valuable feedback. This work is Partially supported by DARPA QuICC, ONR MURI 2024 award on Algorithms, Learning, and Game Theory, Army-Research Laboratory (ARL) grant W911NF2410052, NSF AF:Small grants 2218678, 2114269, 2347322 Max Springer was supported by the National Science Foundation Graduate Research Fellowship Program under Grant No. DGE 1840340. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

References

Abraham, I., Bartal, Y., and Neimany, O. Advances in metric embedding theory. In *Proceedings of the thirty-eighth annual ACM symposium on Theory of computing*, pp. 271–286, 2006.

Arora, S., Lee, J. R., and Naor, A. Euclidean distortion and the sparsest cut. In *Proceedings of the thirty-seventh annual ACM symposium on Theory of computing*, pp. 553–562, 2005.

Arora, S., Rao, S., and Vazirani, U. Expander flows, geometric embeddings and graph partitioning. *Journal of the ACM (JACM)*, 56(2):1–37, 2009.

Aumann, Y. and Rabani, Y. An o (log k) approximate min-cut max-flow theorem and approximation algorithm. *SIAM Journal on Computing*, 27(1):291–301, 1998.

Awerbuch, B. and Azar, Y. Buy-at-bulk network design. In *Proceedings 38th Annual Symposium on Foundations of Computer Science*, pp. 542–547, 1997a. doi: 10.1109/SFCS.1997.646143.

Awerbuch, B. and Azar, Y. Buy-at-bulk network design. In *Proceedings 38th Annual Symposium on Foundations of Computer Science*, pp. 542–547. IEEE, 1997b.

Bartal, Y. Probabilistic approximation of metric spaces and its algorithmic applications. In *Proceedings of 37th Conference on Foundations of Computer Science*, pp. 184–193. IEEE, 1996.

Bartal, Y. Graph decomposition lemmas and their role in metric embedding methods. In Albers, S. and Radzik, T. (eds.), *Algorithms – ESA 2004*, pp. 89–97, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg. ISBN 978-3-540-30140-0.

Bernstein, A. Fully dynamic (2+ ε) approximate all-pairs shortest paths with fast query and close to linear update time. In *2009 50th Annual IEEE Symposium on Foundations of Computer Science*, pp. 693–702. IEEE, 2009.

Bhattacharya, S., Lattanzi, S., and Parotsidis, N. Efficient and stable fully dynamic facility location. *Advances in neural information processing systems*, 35:23358–23370, 2022.

Blum, A., Konjevod, G., Ravi, R., and Vempala, S. Semi-definite relaxations for minimum bandwidth and other vertex-ordering problems. In *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, pp. 100–105, 1998.

Bourgain, J. On lipschitz embedding of finite metric spaces in hilbert space. *Israel Journal of Mathematics*, 52:46–52, 1985.

Charikar, M., Chekuri, C., Goel, A., and Guha, S. Rounding via trees: Deterministic approximation algorithms for group steiner trees and k-median. *Conference Proceedings of the Annual ACM Symposium on Theory of Computing*, pp. 114–123, 1998. ISSN 0734-9025. Proceedings of the 1998 30th Annual ACM Symposium on Theory of Computing ; Conference date: 23-05-1998 Through 26-05-1998.

Chawla, S., Gupta, A., and Räcke, H. Embeddings of negative-type metrics and an improved approximation to generalized sparsest cut. *ACM Transactions on Algorithms (TALG)*, 4(2):1–18, 2008.

Chechik, S. Near-optimal approximate decremental all pairs shortest paths. In *2018 IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS)*, pp. 170–181. IEEE, 2018.

Chechik, S. and Zhang, T. Dynamic low-stretch spanning trees in subpolynomial time. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 463–475. SIAM, 2020.

Church, K. W. Word2vec. *Natural Language Engineering*, 23(1):155–162, 2017.

Dunagan, J. and Vempala, S. On euclidean embeddings and bandwidth minimization. In *International Workshop on Randomization and Approximation Techniques in Computer Science*, pp. 229–240. Springer, 2001.

Dütting, P., Fusco, F., Lattanzi, S., Norouzi-Fard, A., and Zadimoghaddam, M. Fully dynamic submodular maximization over matroids. In *International Conference on Machine Learning*, pp. 8821–8835. PMLR, 2023.

Fakcharoenphol, J., Rao, S., and Talwar, K. A tight bound on approximating arbitrary metrics by tree metrics. In *Proceedings of the Thirty-Fifth Annual ACM Symposium on Theory of Computing*, STOC '03, pp. 448–455, New York, NY, USA, 2003. Association for Computing Machinery. ISBN 1581136749. doi: 10.1145/780542.780608. URL https://doi.org/10.1145/780542.780608.

Feige, U. Approximating the bandwidth via volume respecting embeddings. In *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, pp. 90–99, 1998.

Forster, S. and Goranci, G. Dynamic low-stretch trees via dynamic low-diameter decompositions. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*, pp. 377–388, 2019.

Forster, S., Goranci, G., and Henzinger, M. Dynamic maintenance of low-stretch probabilistic tree embeddings with applications. In *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pp. 1226–1245. SIAM, 2021.

Forster, S., Goranci, G., Nazari, Y., and Skarlatos, A. Bootstrapping dynamic distance oracles. *arXiv preprint arXiv:2303.06102*, 2023.

Garg, N., Konjevod, G., and Ravi, R. A polylogarithmic approximation algorithm for the group steiner tree problem. *Journal of Algorithms*, 37(1):66–84, 2000. ISSN 0196-6774. doi: https://doi.org/10.1006/jagm.2000.1096. URL https://www.sciencedirect.com/science/article/pii/S0196677400910964.

Henzinger, M., Krinninger, S., and Nanongkai, D. Decremental single-source shortest paths on undirected graphs in near-linear total update time. *Journal of the ACM (JACM)*, 65(6):1–40, 2018.

Indyk, P. Algorithmic applications of low-distortion embeddings. In *Proc. 42nd IEEE Symposium on Foundations of Computer Science*, pp. 1, 2001.

J., M. Lectures on discrete geometry. *Graduate Texts in Mathematics*, 2002. ISSN 0072-5285. doi: 10.1007/978-1-4613-0039-7. URL https://cir.nii.ac.jp/crid/1361981469479209856.

Johnson, W. B., Lindenstrauss, J., and Schechtman, G. Extensions of lipschitz maps into banach spaces. *Israel Journal of Mathematics*, 54(2):129–138, 1986. doi: 10.1007/BF02764938. URL https://doi.org/10.1007/BF02764938.

Kleinberg, J. and Tardos, E. Approximation algorithms for classification problems with pairwise relationships: Metric labeling and markov random fields. *J. ACM*, 49(5):616–639, sep 2002. ISSN 0004-5411. doi: 10.1145/585265.585268. URL https://doi.org/10.1145/585265.585268.

Krauthgamer, R., Lee, J. R., Mendel, M., and Naor, A. Measured descent: A new embedding method for finite metrics. In *45th Annual IEEE Symposium on Foundations of Computer Science*, pp. 434–443. IEEE, 2004.

Larsen, K. G. and Nelson, J. Optimality of the johnson-lindenstrauss lemma. In *2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS)*, pp. 633–638. IEEE, 2017.

Leskovec, J. and Krevl, A. SNAP Datasets: Stanford large network dataset collection. http://snap.stanford.edu/data, June 2014.

Linial, N., London, E., and Rabinovich, Y. The geometry of graphs and some of its algorithmic applications. *Combinatorica*, 15:215–245, 1995.

Roditty, L. and Zwick, U. On dynamic shortest paths problems. In *Algorithms–ESA 2004: 12th Annual European Symposium, Bergen, Norway, September 14-17, 2004. Proceedings 12*, pp. 580–591. Springer, 2004.

Roditty, L. and Zwick, U. Dynamic approximate all-pairs shortest paths in undirected graphs. *SIAM Journal on Computing*, 41(3):670–683, 2012.

Subercaze, J., Gravier, C., and Laforest, F. On metric embedding for boosting semantic similarity computations. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pp. 8–14, 2015.

Thorup, M. and Zwick, U. Approximate distance oracles. *Journal of the ACM (JACM)*, 52(1):1–24, 2005.

A. Empirical validation

We tested the theoretical algorithm guarantees on three different graphs.

Data sets preparation. As the backbone for each graph, we used the social network of LastFM users from Asia available in the Stanford Network Analysis Project dataset (SNAP) (Leskovec & Krevl, 2014). To adhere to our dynamic setting, we randomly chose a subset of 150, 300, and 600 connected nodes to form three different bases of the dynamically changing network. We added random weights from a uniform distribution to these graphs. We augmented each graph by respectively 10000, 5000, and 1000 changes to the topology (queries). Each change increases the weight of a randomly and uniformly chosen edge of the graph by a number chosen from a uniform distribution whose range increases as the process progresses.

Evaluation. We implemented the cut-preserving embedding from Theorem 1.3 and computed the distances between every pair of nodes in the graph after each query. We compared the average of these distances with the average distances computed by an exact algorithm that in an offline fashion computes the shortest distances after each query. Visualized results are presenting in Figure 2. To allow more direct reasoning about the distortion of our embedding, in Figure 3 we provide plots

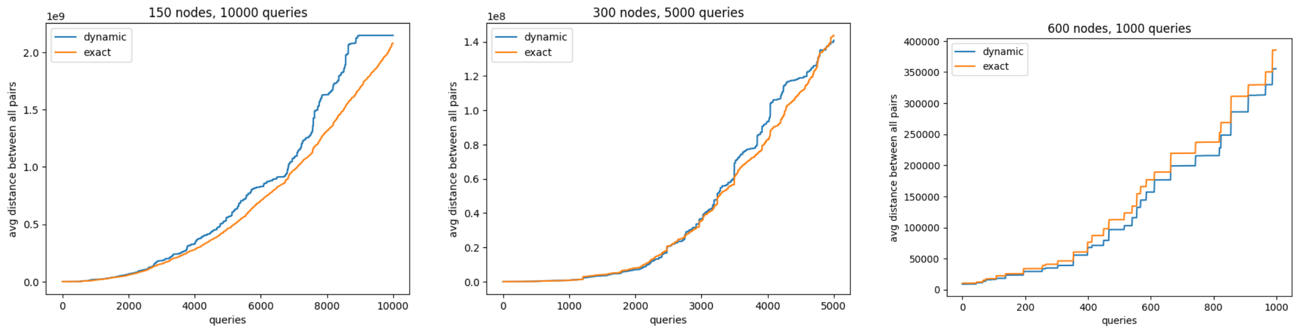

Figure 2. Visualization of average distances in a dynamically changing metric. The orange line represents the average distance between all pairs of nodes computed exactly, using a deterministic shortest path algorithm, after every query. The blue line represents the average distance computed based on the embedding given by the dynamic embedding algorithm proposed in the paper.

representing, after each query, the ratio of the average distance based on our dynamic embedding to the average distance computed exactly. We would like to note that even though theoretically our embedding is not contractive, this property holds in expectation. In practice, small fluctuations may appear which are particularly visible in the case of a small number of queries.

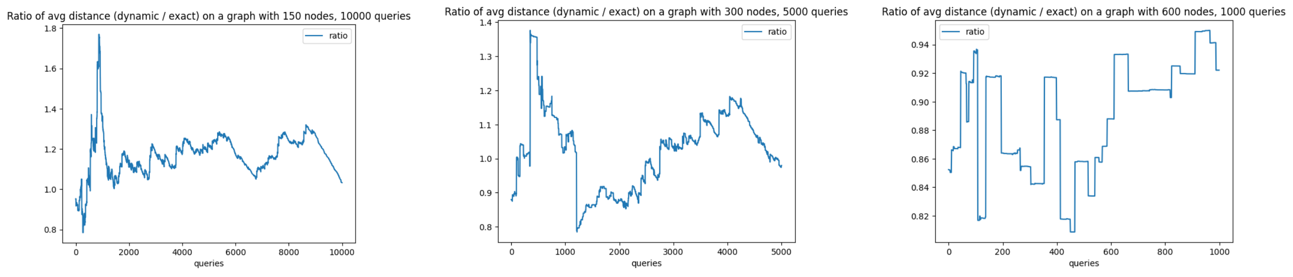

Figure 3. The ratio of the average distance between all pairs of points computed by of our embedding to the exact average distance between all pairs, after each query.

Conclusions. We observed that in these experiments the achieved stretch is within a constant factor of the average over the exact distances which adheres to the $O(log^2(n))$ theoretical bound of Theorem 1.3 and even surpasses it. This advantage might be a consequence of a couple of things, e.g. the random process involved in the generation, or a few number of testing instances. Nevertheless, we find this results promising. We hope that they can serve as a starting point for further investigation of practicality of dynamic embedding.

B. Omitted Proofs

B.1. Static Algorithm

We briefly provide a sketch of for the proof of Theorem 4.5 below and refer to Bartal (Bartal, 1996) for the full details. The main addendum we invoke to this standard approach is the edge contraction stage to ensure "close" nodes are not segregated out at a decomposition stage.

Proof sketch. We construct the so-called "low diameter randomized decomposition" (LDRD) for the input graph G with (parameter R) via the following high-level procedure: first, contract all paths between vertices that are shorter than $\frac{R}{2n}$. Now, starting with the contracted graph we pick an arbitrary vertex, u, and select a radius r sampled from the geometric distribution with success parameter β. Mark all unmarked vertices contained within the set $B_r(u) = \{v \in G : d(u, v) \leq r\}$ and define this to be a new cluster of the partition. Repeat this ball-growing process with the next unmarked vertex in G and only consider the remaining unmarked vertices to be included in future clusters. This procedure is repeated until all vertices are marked and return the created clusters. Pseudocode for this is provided in Algorithm 1.

To verify the theorem, we proceed to prove each property of a (β, R)-weak decomposition holds with the additional probabilistic bound. The second property holds by contraction of small edges. Therefore, we need only obtain an upper bound on the probability that the two vertices u and v are not assigned to the same cluster, ie., $C(u) \neq C(v)$. This final point is a standard procedure attributed to (Bartal, 1996) which we overview below.

For some edge $e = (u, v)$, we need to bound the probability that e is contained within none of the clusters of the decomposition. Specifically, we need to ensure that after each ball growing procedure (Line 8 of Algorithm 1) e is not added to a cluster. We can therefore decompose this value as the probability that exactly one end point of e was contained in the last ball grown or neither is contained. Let t denote the stage of the ball growing procedure and let X_t denote the event that edge $(u, v) \notin C_j$ for all $j \leq t$. Then we can recursively compute $\Pr[X_t]$ as the probability that exactly one of u, v are contained in C_t, or the probability that neither is as a condition on $\Pr[X_{t+1}]$. The direct computation of these values using the probability density functions for the radius random variable's distribution and bounding the probability that an endpoint of $(u, v) \in C_t$ is contained in Section 3 of (Bartal, 1996) and yields the desired $O\left(\frac{\log n}{R}\right)$ upper bound on the probability that $C(u) \neq C(v)$.

□

Proof of Lemma 4.6. By the guarantee on the partition given by Theorem 4.5, for every pair of vertices $u, v \in V$ we must have that $\Pr[u \text{ and } v \text{ in different clusters}] \leq \beta \cdot d(u, v)$, and $\Pr[u \text{ and } v \text{ in different clusters}] = 0$ for all u, v such that $d_G(u, v) < \frac{R}{n}$. Note that if the two nodes are in the same cluster than they must be on the same side of the cut. Therefore, the first two properties of (β, R)-cuts are proved.

Now, further assume that $u, v \in V$ such that $d(u, v) > R$. By construction, for all C_i, have $\text{wdiam}(C_i) \leq R$. Therefore, the vertices u and v cannot be contained in the same cluster. Let C_u and $C_v \neq C_u$ denote the clusters containing u and v respectively. By construction of the cut set S, we have that $\Pr[\texttt{cut}_{C_u, C_v}] = \frac{1}{2}$. Therefore, $\Pr[\texttt{cut}_{u,v}] \geq 1/2$ as claimed. □

Proof of Theorem 4.8. We divide the proof into a few key lemmas. For every pair of vertices u, v, define $d'_p(u, v) := \|\rho(u) - \rho(v)\|_p$. To begin, we demonstrate that, in expectation, the characteristic embedding is non-contractive.

Claim 1. For every pair of vertices $u, v \in V$:

$$\mathbb{E}\left[d'_p(u, v)\right] \geq \mathbb{E}\left[d'_\infty(u, v)\right] \geq d(u, v) \, .$$

Proof. The first inequality holds for any embedding ρ by the triangle inequality, and we therefore focus on the second inequality. For every fixed u, v, consider the maximal i such that $2^i \leq d(u, v)$. Such an i exists given the assumption $\Delta \geq \text{diam}(G)$. By definition of a distance preserving cut, we have $\Pr[\texttt{cut}_{u,v}(S_i)] \geq \frac{1}{2}$ which implies that with probability at least $\frac{1}{2}$, the i-th coordinate $\rho(u)$ and $\rho(v)$ different. This implies the claim by the definition of the characteristic embedding.

Formally,

$$\begin{aligned}\mathbb{E}\left[d'_\infty(u,v)\right] &\geq \mathbb{E}\left[|\,\rho_i(u)-\rho_i(v)\,|\right] \\ &\geq R_i \cdot \Pr\left[|\,\rho_i(u)-\rho_i(v)\,| \geq R_i\right] \\ &= R_i \cdot \Pr\left[\mathtt{cut}_{u,v}(S_i)\right] \\ &= \frac{R_i}{2} \geq \frac{d(u,v)}{4}\end{aligned}$$

thus, proving the claim. □

We proceed to verify that, in the ℓ_1-metric, the distance between any two embedded points is only distorted by a polylogarithmic factor which further bounds the ℓ_p norm by such an approximation.

Claim 2. For every pair of vertices $u, v \in V$:

$$\mathbb{E}\left[d'_1(u,v)\right] \leq O(\log^2 n) \cdot d(u,v)\ .$$

Proof. For convenience, we denote $d := d(u,v)$. Note that, for every scale i, we have

$$\begin{aligned}\mathbb{E}\left[\|\rho_i(u)-\rho_i(v)\|_1\right] &= \mathbb{E}\left[R_i \cdot \mathbb{1}\left\{\mathtt{cut}_{u,v}(S_i)\right\}\right] \\ &= R_i \cdot \Pr\left[\mathtt{cut}_{u,v}\right].\end{aligned} \tag{2}$$

By definition, the ℓ_1 norm will merely be a summation on the above over the scales of R_i:

$$\begin{aligned}\mathbb{E}\left[\|\rho(u)-\rho(v)\|_1\right] &= \sum_{i=1}^{\log\Delta} \mathbb{E}\left[|\rho_i(u)) - \rho_i(v)|\right] \\ &= \sum_{i=1}^{\log\Delta} R_i \cdot \Pr\left[\mathtt{cut}_{u,v}\right].\end{aligned}$$

We proceed to bound this summation by bounding individual summands corresponding to manageable scales with the properties of our distance preserving cuts from Section 4.1.

First observe that for every i such that $2^{i+2} < d$, we must have that a ball of radius $R_i = 2^{i+1}$ cannot contain both points of the embedding. Therefore, $|\rho_i(u) - \rho_i(v)| = R_i$. It follows that

$$\sum_{i=1}^{\log(d)} \mathbb{E}\left[\rho_i(u)-\rho_i(v))\right] \leq \sum_{i=1}^{\log(d)} 2^i \leq 4 \cdot d\ .$$

Moreover, by Theorem 4.5 we have that $nd < R_i$ implies u and v are embedded to the same cluster which implies their embedded ℓ_1 distance for the coordinate corresponding to this cluster does not contribute to the overall distortion. Lastly, combining these facts with Eq. 2 we have that $\mathbb{E}\left[\|\rho(u)-\rho(v)\|_1\right]$ is equivalent to

$$\begin{aligned}
&\sum_{i=1}^{\log(d)} \mathbb{E}\left[|\rho_i(u)-\rho_i(v))|\right] + \ldots \\
&+ \sum_{i=\log(d)}^{\log(nd)} \mathbb{E}\left[|\rho_i(u)-\rho_i(v))|\right] + \ldots \\
&+ \sum_{i>\log(nd)}^{\log\Delta} \mathbb{E}\left[|\rho_i(u)-\rho_i(v))|\right] \\
&\le \sum_{i=1}^{\log(d)} 2^i + \sum_{i=\log(d)}^{\log(nd)} R_i \cdot \Pr\left[\mathtt{cut}_{u,v}\right] + \sum_{i>\log(nd)}^{\log\Delta} 0 \\
&\le 4\cdot d + \sum_{i=\log(d)}^{\log(nd)} R_i \cdot \beta_i \cdot d \\
&= 4\cdot d + \sum_{i=\log(d)}^{\log(nd)} O(\log n)\cdot d\,,
\end{aligned}$$

where the final inequality comes from our definition of β_i. We lastly bound the remaining term using the observation that

$$\sum_{i=\log(d)}^{\log(nd)} O(\log n)\cdot d = O(\log^2 n)\cdot d\,,$$

which completes the proof of the claim. □

Combining the two claims, we have the result of Theorem 4.8. □

B.2. Dynamic Algorithm

Proof of Theorem 5.1. The decremental algorithm uses $\log(\Delta)$ instances of algorithm $\mathcal{A}$ with the given choice of parameters which allows us to follow the argument provided for the static case in Section 4. Let $R_i = 2^i$ for $1 \le i \le \log(\Delta)$. The i-th instance of the algorithm $\mathcal{A}$ maintains a $\left(\frac{\log n}{R_i}, R_i, \frac{R_i}{n}\right)$-distance preserving cut S_i. The final embedding $\rho : V \to \mathbb{R}^{\log\Delta}$ is the characteristic embedding of the cuts $(S_1, \ldots, S_{\log(\Delta)})$ and parameters $\left(R_1, \ldots, R_{\log(\Delta)}\right)$. Formally, we set the i-th coordinate of $\rho(v)$ to be R_i if $v \in S_i$ and to 0 otherwise.

Upon the arrival of a decremental change, the algorithm inputs this change into every instance of the dynamic decremental algorithm $\mathcal{A}$ it runs. By assumption on the input algorithm $\mathcal{A}$, each run explicitly outputs changes to the structure of the i-th cut. Therefore, the main algorithm can adapt to these changes by appropriately changing the coordinates of vertices. Specifically, if a vertex is removed from the i-th cut its coordinate changes from 0 to R_i and if it is added the opposite change takes place. Since processing each such change takes constant time, and there are $t(m,n)$ updates in total by assumption on $\mathcal{A}$, the total time for the i-th instance is $t(m,n)$. Therefore, by charging the time used for these changes to the total update time of $\log\Delta$ instances of the algorithm $\mathcal{A}$, the total running time of the decremental algorithm is at most $O\left(t(m,n)\log\Delta\right)$. Since there are $\log(\Delta)$ instances of the algorithm, it follows that the total update time is $O(t(m,n)\log(\Delta))$. As for the distortion, by assumption on algorithm $\mathcal{A}$, each run maintains a $\left(\frac{\log n}{R_i}, R_i, \frac{R_i}{n}\right)$-distance preserving cut. Thus, the stretch of the maintained embedding ρ follows from applying Theorem 4.8. □

Proof sketch of Lemma 5.2. At a high level, the dynamic algorithm presented in (Forster et al., 2021) for maintaining a weak decomposition undergoing edge deletions relies on the concept of assigning a center to each cluster in the decomposition, an idea initially introduced in (Chechik & Zhang, 2020).

This technique employs a dynamic Single Source Shortest Paths (SSSP) algorithm (specifically, the SSSP algorithm described in (Henzinger et al., 2018)) to monitor the distances from a center to every vertex within the cluster. Whenever an edge is deleted, the change is also updated to the SSSP algorithm. The SSSP algorithm then outputs vertices from the cluster whose distance to the center is greater than a certain threshold, and such that keeping these vertices within the cluster could potentially violate the requirements of a (β, δ)-weak decomposition. To prevent this event, an appearance of such a vertex incurs either a re-centering of the current cluster or splitting the cluster into two or more disjoint new clusters. These operations ensure that eventually the diameter of each cluster satisfies the requirements of the (β, R)-weak decomposition. Crucially, the authors show that the number of times a cluster is re-centered can be bounded by $a \log n$, for some absolute constant a. On the other hand, the splitting procedure is designed in such a way that the size of a cluster that is split from the previous cluster shrinks by at least a factor of 2. As a result, any vertex can be moved to a new cluster at most $O(\log n)$ times.

Now, the crucial observation that allows us to carry the approach to the case of edge weight increases is the fact that the SSSP algorithm of (Henzinger et al., 2018) also handles edge weight increases while preserving the same complexity bounds. The algorithm then is the same as in (Forster et al., 2021) with the only change that, to monitor distance from a center of a cluster to every other vertex, we use the SSSP version that supports edge weights increases. As for the correctness part, we can carry the analysis from (Forster et al., 2021) with minor changes. In Lemma 3.3, we show that the probability of being an inter-cluster edge is at most $\beta w_k(e)$, where $w_k(e)$ denotes the weight of edge e in the current dynamic graph G_k after k updates. This, by the union bound, implies that for every pair of vertices $u, v \in G_k$ it holds $\Pr\left[C(u) \neq C(v)\right] \leq \beta \cdot d_{G_k}(u, v)$. From the fact that weight updates only increase distances, we observe that Lemmas 3.5 and 3.6 from (Forster et al., 2021) still hold (here, it is also important that updates are independent of the algorithm, which is also true in our model). This observation ultimately gives us that each cluster undergoes a center re-assignment at most $O(a \cdot \log n)$ times. As a consequence, our derivation of total update time follows from the one in (Forster et al., 2021) with the change that we account $O(Q)$ time to parse all updates. □

Proof of Lemma 5.3. The algorithm starts by preprocessing graph G. First, replaces all edge weights of a value smaller than ϵ with a weight of 0. Since the underlying ball-growing processes responsible for clustering in Lemma 5.2 samples radii of clusters from a continuous distribution, we have that with probability 1 all vertices are connected by a path of weight 0 are in the same cluster. This in turn implies that if $d(u, v) < \epsilon$ then $C(u) = C(v)$. This preprocessing step takes $O(m)$ time.

As a next step, the algorithm initializes the dynamic decremental algorithm from Lemma 5.2 for the choice of parameters $\left(\frac{\log n}{R}, R\right)$, $a = O(1)$ on the preprocessed graph G. It can be easily checked that this choice of parameters satisfies the assumptions of the lemma. As a consequence of the initialization, a $\left(\frac{\log n}{R}, R\right)$-weak decomposition of the preprocessed G is computed. Denote the clusters of this decomposition $C_1, \ldots, C_k$. The algorithm then samples k uniform and independent values from $\{0, 1\}$, one value for each cluster. Next, a cut of G is created by grouping vertices from clusters that have been assigned value 1, denoting this side of the cut S. By Lemma 4.6, we have that the cut S is a $\left(\frac{\log n}{R}, R\right)$-distance preserving cut. We also note that the cut S can be generated with $O(k) = O(n)$ additive overhead to the time complexity of the dynamic decremental algorithm from Lemma 5.2.

Finally, we discuss the algorithm action upon a decremental update of an edge. Whenever there is an edge weight increase, the algorithm inputs this change to the algorithm that dynamically maintains $\left(\frac{\log n}{R}, R\right)$-weak decomposition of G. According to Lemma 5.2, the only changes to the partitioning occur when splitting an existing cluster into two or more new clusters and members of the new clusters need to be explicitly listed. Let $\mathcal{C}' = \{C'_1, \ldots, C'_j\}$ be the set of these newly formed clusters. The main algorithm temporarily deletes the vertices belonging to clusters $\mathcal{C}'$ from the dynamic cut S it maintains. Then, for each new cluster, it samples independently and uniformly a value from $\{0, 1\}$. Again, vertices from clusters that have been sampled 1 are added to those who already had 1 (before the decremental update), while the ones who have just sampled 0 are assigned to the other side of the cut. Since the decremental change is oblivious to the randomness of the algorithm, the random bits assigned to all clusters of the new partitioning obtained from the decremental update are distributed i.i.d. according to a Bernoulli distribution with parameter $\frac{1}{2}$, and by applying Lemma 4.6 we obtain that the updated cut is also a $\left(\frac{\log n}{R}, R\right)$-distance preserving cut. As for the time complexity, we can observe that the number of changes made to the structure is linearly proportional to the number of vertices changing a cluster after an update. It, therefore, follows that the total update time needed for maintaining the cut is dominated by the time needed to report the changes in the dynamic partitioning, and according to Lemma 5.2, is at most $O\left(m^{1+o(1)} \log^2 W + Q\right)$. □