Dynamic Metric Embedding into ℓ_p Space

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

[1]University of Maryland, College Park, USA [2]Augusta University, Georgia, USA

Metric Embedding

An *embedding* of a metric space (X, d_X) into a metric space (Y, d_Y) is a function $f : X \to Y$ such that for every pair of points $x, y \in X$, the distance $d_X(x, y)$ between x and y in X is closely preserved in Y under f.

The *distortion* of an embedding $f : X \to Y$ is a measure of how much the embedding f stretches or compresses distances between points. It is formally defined as:

$$\text{distort}(f) = \sup_{x,y\in X, x\neq y} \frac{d_Y(f(x), f(y))}{d_X(x,y)} \cdot \sup_{x,y\in X, x\neq y} \frac{d_X(x,y)}{d_Y(f(x), f(y))}$$

where the first term is called the *expansion* and the second term is called the *contraction* of the embedding.

- Facilitates dimensionality reduction while maintaining relative distances within a multiplicative factor (distortion).
- Essential for simplifying complex network optimization problems and designing efficient approximation algorithms, particularly useful for the sparsest cut problem.

Dynamic Algorithms

An algorithm *maintains an embedding* of a graph G into a metric space (X, d_X) dynamically if it continuously updates an embedding $\phi : V(G) \to X$ in response to changes in G. For each update in G, the algorithm adjusts ϕ to reflect these changes while trying to minimize the distortion of the original embedding. The process is defined formally as follows:

- **Dynamic Model:** The graph G undergoes a sequence of updates, which may include additions, deletions, or modifications of edges or vertices.
- **Update Protocol:** After each update, the algorithm outputs the new mapping ϕ_t for each vertex v whose embedding has changed, where t indexes the sequence of updates.
- **Objective:** The goal is to maintain the quality of the embedding ϕ, such that the distortion between any two points $u, v \in V(G)$ in the embedding space X remains within acceptable bounds, ideally minimizing any increases in distortion due to updates.

Prior Results

- **Bourgain** [2]: Embed into any ℓ_p space with logarithmic distortion.
- **Johnson and Lindenstrauss** [4]: Embed into Euclidean space with $1 + \varepsilon$ distortion
- **Bartal** [1]: Embed into distributions of trees with logarithmic distortion, introduced ball growing technique
- **Forster et al.** [3]: Dynamic ball growing which can handle edge insertions or deletions.

The Forster et al. dynamic ball maintenance is a key subroutine used by our algorithm.

Main Contributions

(1) We show that there is no fully dynamic algorithm that can explicitly maintain a dynamic embedding into ℓ_p space with high probability.

(2) We introduce a dynamic algorithm that achieves an embedding with expected distortion $O(\log^2 n)$, where n is the number of vertices in G. This embedding is efficiently maintained over a sequence of updates with total update time $O((m^{1+o(1)} \log^2 W + Q) \log(nW))$, where W is the maximum edge weight, Q is the total number of updates, and m is the number of edges.

The full version of our paper can be found on arXiv with the QR-code above.

Problem Formulation

Given a weighted, undirected graph G undergoing a sequence of edge weight updates, the objective is to maintain a dynamic embedding of G into a normed space (X, ℓ_p) such that for every pair of vertices u and v in G, the distance between $\phi(u)$ and $\phi(v)$ in X approximates the graph distance $d_G(u, v)$ in G with a bounded distortion factor.

- Maintain a mapping $\phi : G \to X$ such that the expected distortion of distances between any two vertices $u, v \in G$ is minimized.
- The updates to the embedding ϕ should be efficient in response to changes in the graph G due to edge weight modifications.

Impossibility Results

We first show that maintaining an embedding undergoing edge weight increases and decreases (fully dynamic) to the original graph yields unbounded distortion. We thus focus on the decremental setting.

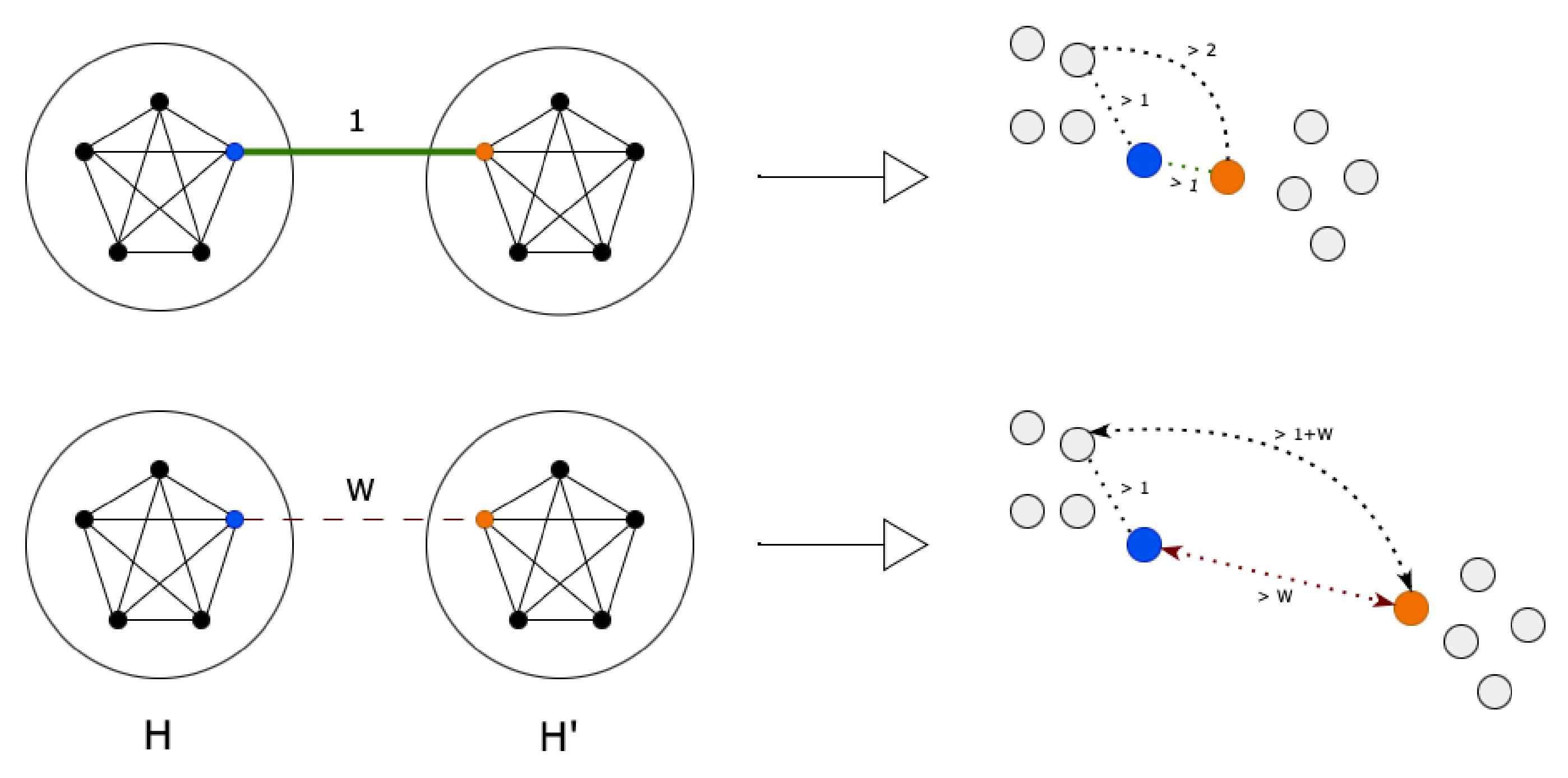

Figure 1. Adversarial sequence of graph updates

Algorithm Overview

Our algorithm dynamically maintains a low-distortion embedding of a graph into ℓ_p space by efficiently processing edge weight updates. It employs a novel combination of randomized decomposition and clustering to handle changes dynamically while ensuring the embedding's fidelity.

Initial Embedding / Static Decomposition: Utilizes a static randomized decomposition to create an initial embedding of the graph into ℓ_p space, setting a baseline with minimized initial distortion.

Dynamic Updates:

- Adjusts embeddings incrementally in response to edge weight modifications, maintaining the embedding's integrity and quality. Leverage [3] tools to maintain partitions from static part.
- Recalculates only the affected segments of the embedding, split clusters efficiently to maintain distortion bound without recomputing decomposition.

Key Takeaways

- Introduced the first dynamic embedding into ℓ_p space that accommodates edge weight increases, marking a significant advancement over static methods.
- Achieves polylogarithmic distortion with update times competitive with the best-known algorithms for related problems.
- Validates the practicality of dynamic embeddings for handling real-time updates in complex networks, potentially transforming approaches to network analysis and optimization.

Future Research: Opens avenues for further refinement in dynamic algorithm design and its application to other complex data structures and optimization problems.

Experimental Results

Data Set Preparation: Utilized the LastFM Asia social network from the Stanford Network Analysis Project dataset (SNAP) [5]. Constructed three graphs from subsets of 150, 300, and 600 nodes, with edge weights assigned from a uniform distribution. Graphs underwent 10,000, 5,000, and 1,000 dynamic updates respectively, with each update increasing the weight of a randomly selected edge.

Evaluation: Implemented the dynamic cut-preserving embedding. Calculated distances between node pairs post each update and compared these with exact distances computed offline. Results visualized in Figure 2.

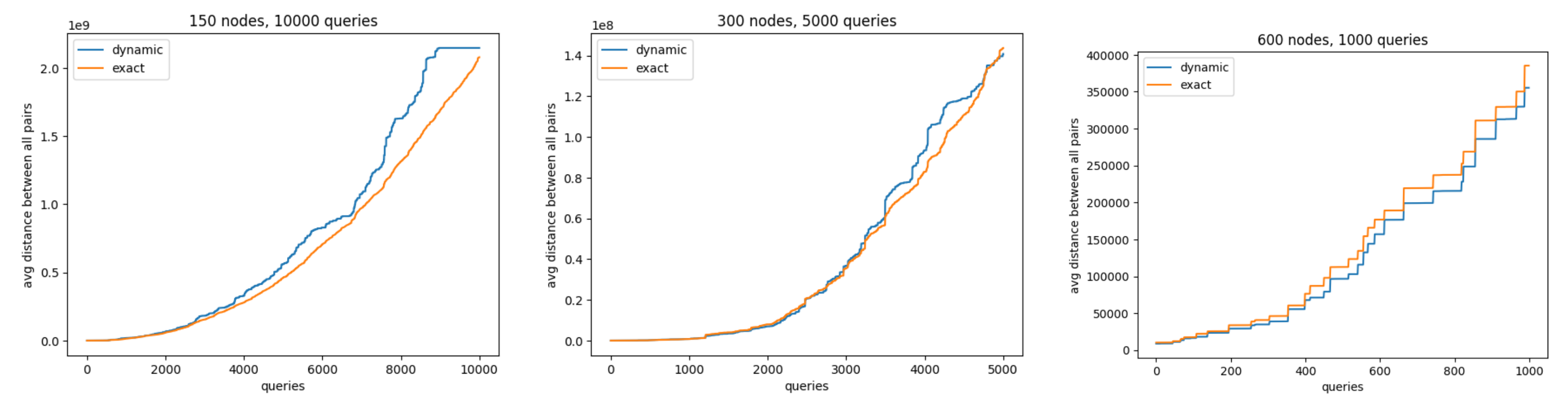

Figure 2. Visualization of average distances in a dynamic metric. The orange line shows the exact average distance between nodes, computed using a deterministic shortest path algorithm after each query. The blue line illustrates the average distance from the proposed dynamic embedding algorithm.

Figure 3 presents plots of the ratio of the average distance from our dynamic embedding to the exact average distance after each query, facilitating direct assessment of embedding distortion. Note that while the embedding is theoretically non-contractive, it achieves expected contractiveness. Practical results may show minor fluctuations, especially with fewer queries.

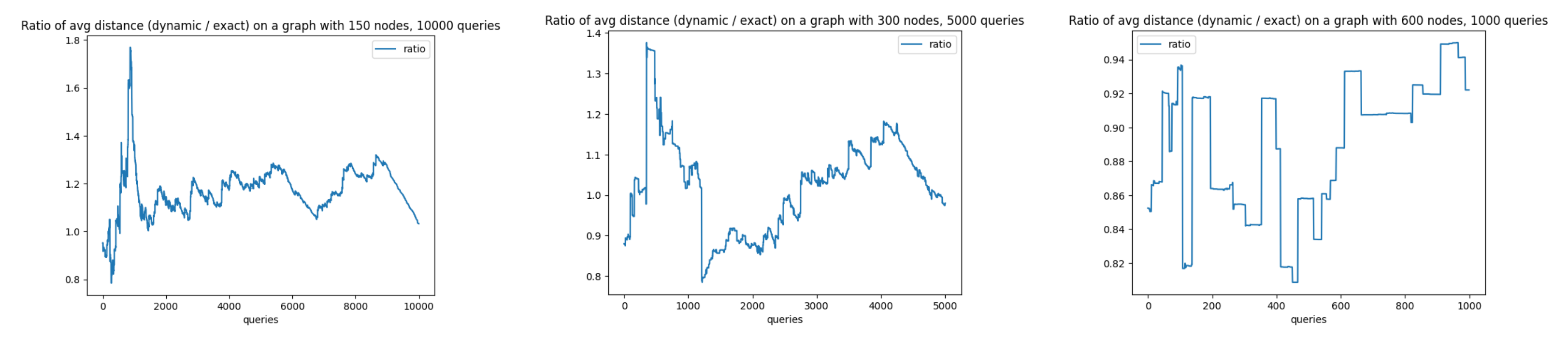

Figure 3. The ratio of the average distance between all pairs of points computed by of our embedding to the exact average distance between all pairs, after each query.

Acknowledgements

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

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

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

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

[4] W. B. Johnson, J. Lindenstrauss, and G. Schechtman. Extensions of lipschitz maps into banach spaces. *Israel Journal of Mathematics*, 54(2):129–138, 1986.

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