Ben-Gurion University of the Negev

Thesis submitted in partial fulfillment of the requirements for the degree of “DOCTOR OF PHILOSOPHY”

On Refined Notions of Embeddings

By

Arnold Filtser

Submitted to the Senate of Ben-Gurion University of the Negev

March 2019

Beer-Sheva

Ben-Gurion University of the Negev

Thesis submitted in partial fulfillment of the requirements for the degree of "DOCTOR OF PHILOSOPHY"

On Refined Notions of Embeddings

Author:

Arnold Filtser

Submitted to the Senate of Ben-Gurion University of the Negev

Approved by:

Professor
Robert Krauthgamer
Advisor

Professor
Ofer Neiman
Advisor

March 2019

Beer-Sheva

This work was carried out under the supervision of

Prof. Robert Krauthgamer and *Prof. Ofer Neiman*

In the Department of Computer Science
Faculty of Natural Sciences

I Arnold Filtser, whose signature appears below, hereby declare that

- I have written this thesis by myself, except for the help and guidance offered by my thesis advisors.
- The scientific materials included in this thesis are products of my own research, culled from the period during which I was a research student.
- This thesis incorporates research materials produced in cooperation with others. Specifically:
  The results in Part II were obtained jointly with Michael Elkin and Ofer Neiman.
  The results in Part III were obtained jointly with Yair Bartal and Ofer Neiman.
  The results in Part V were obtained jointly with Robert Krauthgamer.

26-Mar-2019	Arnold Filtser	
Date	Name	Signature

Acknowledgements

First and foremost I would like to thank the Israeli taxpayer who supported my research via various funds. Specifically, this research was supported in part by the planning and budgeting committee Levtzion scholarship, Kreitman School Negev Scholarship, general BGU CS department funds, ISF grant No. (523/12 and 1817/17) and BSF Grant 2015813. I would like to thank the many individuals and organizations for their private donations to BGU, the fruits of which I also enjoyed. Further, I would like to thank the European taxpayer who supported me via the European Union Seventh Framework Program (FP7/2007-2013) under grant agreement $n^{\circ}303809$. In addition I would like to thank for various travel funds I received during the PhD (in particular from SIAM, ACM, Microsoft, Google).

Next, I would like to express my special appreciation and thanks to my advisers, Professors Robert Krauthgamer and Ofer Neiman. I learned a lot from you both. Robi, you are an endless source of knowledge, always full with ideas, possible directions and good advice, thanks a lot! Ofer, thank you for the patience, continuous support, all the hours of deep discussions, constant availability and great ideas. I have a lot of gratitude, this PhD was a real pleasure. Additionally, I would like to thank Professor Michael Elkin. Thank you for suggesting many excellent research problems, and often sharing various advice with me.

I gratefully acknowledge all my collaborators: Ittai Abraham, Stephen Alstrup, Yair Bartal, Shiri Chechik, Søren Dahlgaard, Michael Elkin, Omrit Filtser, Lee-Ad Gottlieb, Anupam Gupta, Robert Krauthgamer, Ofer Neiman, Shay Solomon, Morten Stöckel, Nimrod Talmon, Ohad Trabelsi and Christian Wulff-Nilsen. It was a delight to work with you, and I hope for future collaborations.

I also thank Ben-Gurion University, for its funding, flexibility and thoughtfulness. Many thanks also to the the technical and administrative staff who make this research pleasant and possible. Thanks to all the people who attend and organize the theory seminar, for the excellent atmosphere.

Special thanks to the Weizmann institute for our extra-marital relationship... Thanks for the extreme flexibility, the excellent courses, and the theory lunch. The place is simply buzzing with ideas. Additionally, thanks to all the members of Robi's group throughout the years: Chen Attias, Diptarka Chakraborty, Rajesh Chitnis, Shaofeng Jiang, Lior Kamma, Bundit Laekhanukit, Yevgeny Levanzov, Yosef Pogrow, David Reitblat, Havana Rika, Roi Sinoff, Nimrod Talmon, Ohad Trabelsi, Otniel van Handel and Shai Vardi. Thank you for the outstanding atmosphere, the discussions and all the fun hours spent together.

A special thanks to my family. Words cannot express how grateful I am to my mother and father. Thank you for accepting that my work will not have any practical value in this world or the next... Thank you for encouraging and nurturing my love for math and knowledge, for investing time and money in my education, believing in my abilities and pushing me towards excellence. Thanks a lot also to my siblings, grandparents and all further family for love and support.

Most of all, I would like to express my gratitude and appreciation to my beloved wife Omrit to whom I am deeply obligated for her love, care, infinite support, encouragement, and for always being there. Additionally, for hearing and debating on various ideas, for editing my English, good advice, and for preventing me from becoming an engineer. I would like to thank my daughters Naama and Hadass, for their patience to hear research discussions during dinner. You are an endless source of joy and happiness, a real inspiration.

Finally, I would like to express thanks, respect and admiration to the theoretical computer science community. Thank you for the good will and the collaborative atmosphere. I am really happy to be part of this community.

To Omrit, Naama and Hadass.

Contents

Abstract

Metric spaces are used to represent various relations such as transportation cost between cities or dissimilarity between bacterial strains. However, general metrics can be quite difficult to manage. It would be very convenient if we could represent the points in a more structured form. Such a representation can provide insight and allow us to execute efficient algorithms. For example, we would like to represent the metric as points in Euclidean space, where the distance between every pair of metric points is equal to the Euclidean distance between their representations. Unfortunately, this is impossible. Nevertheless, if we allow to somewhat distort the distances, such a representation becomes possible, and has been found extremely useful.

Embedding is a mapping between metric spaces that approximately preserves the geometry of the original space. Often the host space has a simple structure or desirable features. This wide framework provides an algorithmic methodology, which has been successfully applied for approximation/online/distributed algorithms, etc. However, this methodology appears to have some limitations: the performance inherently depends on the cardinality of the metric. The guarantee is a worst-case type, i.e., the same for all the point pairs. One could not specify in advance which points should enjoy a better service (i.e. distortion, dimension, etc.) than that given by the worst-case guarantee.

We alleviate this limitation by devising a suite of *prioritized* distortion. We show that given a priority ordering $(x_1, x_2, \ldots, x_n)$ of the metric points, one can devise an embedding, in which the distortion incurred by any pair containing a vertex x_j will depend on the rank j of the vertex. The worst-case performance of our embeddings is typically asymptotically no worse than that of their non-prioritized counterparts.

We also study *scaling* distortion, which requires that for every $0 < \epsilon < 1$, the distortion of all but an ϵ-fraction of the pairs is bounded by the appropriate function of ϵ. Such distortion guarantee implies bounds on the average distortion, as well as on higher moments of the distortion function. We show an equivalence theorem between prioritized distortion and a strong version of scaling distortion. This equivalence implies many new embeddings results. Another application is an algorithm that, given weighted undirected graph, returns a *spanning tree* whose weight is at most $(1+\rho)$ times that of the MST, and provids *constant average distortion* $O(1/\rho)$.

We also study the *Steiner Point Removal* problem. Here we are given a weighted graph $G = (V, E)$ and a set of terminals $K \subset V$ of size k. The objective is to find a minor M of G with only the terminals as its vertex set, such that distances between the terminals are preserved up to a small multiplicative distortion. The underlying question would be to consider some restricted graph family. Is it possible to significantly enrich the various geometries induced by k-vertex graphs in the family by adding additional Steiner vertices? Our contribution is an upper bound of $O(\log k)$ on the distortion, improving a previous $O(\log^2 k)$ upper bound. We achieve this upper bound using a novel algorithm called the `Relaxed-Voronoi` algorithm, which is simpler than previously used algorithms. In particular we provide an almost linear time implementation.

Finally we turn to study sparsification. A valued constraint satisfaction problem (VCSP) instance (V, Π, w) is a set of variables V with a set of constraints Π weighted by w. Given a VCSP instance, we are interested in a re-weighted sub-instance $(V, \Pi' \subset \Pi, w')$ that preserves the value of the given instance (under every assignment to the variables) within factor $1 \pm \epsilon$. A well-studied special case is cut sparsification in graphs, which has found various applications. We show that a VCSP instance consisting of a single boolean predicate $P(x, y)$ (e.g., for cut, $P = \mathsf{XOR}$) can be sparsified into $O(|V|/\epsilon^2)$ constraints if and only if the number of inputs that satisfy P is anything but one (i.e., $|P^{-1}(1)| \neq 1$). Furthermore, this sparsity bound is tight unless P is a relatively trivial predicate. We conclude that systems of 2SAT (or 2LIN) constraints can also be sparsified.

Part I

Introduction, Results and Discussion

1 Introduction

Low-distortion metric embeddings are a crucial component in the modern algorithmist toolkit. Given a pair of (finite) metric spaces $(X, d_X), (Y, d_Y)$, a map $\phi : X \to Y$, the *contraction* and *expansion* of the map ϕ are the smallest τ, ρ, respectively, such that for every pair $x, y \in X$,

$$\frac{1}{\tau} \leq \frac{d_Y(\phi(x), \phi(y))}{d_X(x,y)} \leq \rho \quad .$$

The *distortion* of the embedding is $\tau \cdot \rho$. If $\tau = 1$ (resp. $\rho = 1$) we say that the embedding is non-contractive (expansive). If $\rho = O(1)$, we say that the embedding is *Lipschitz*. If $\tau \geq 1$ we say that the embedding is *dominating*. If the distortion $\tau \cdot \rho$ is 1, we say that X embeds *isometrically* into Y.

Metric embeddings have applications in approximation algorithms [LLR95], online algorithms [BBMN11], distributed algorithms [KKM+12], and for solving linear systems and computing graph sparsifiers [ST04a]. The basic approach behind most of the applications is as following: Suppose we have some hard problem in a metric space X. In many cases this problem might become simpler if we assume that X has certain properties (e.g., Euclidean space, tree metric). Suppose further that there is an embedding ϕ of X into a metric space Y that possesses the desired property with distortion t. Instead of solving the problem directly in X, we start by solving the problem efficiently in the embedded space $\phi(X)$. We would then pull the solution back to X, while paying some approximation factor $f(t)$ w.r.t. the optimal solution.

Metric embeddings are often very useful for graphs. Consider a weighted graph $G = (V, E, w)$, the metric d_G associated with the graph is the shortest path metric. Here the distance between a pair of vertices v, u is the weight of the shortest path between them. In the rest of the introduction we describe various results in metric embeddings theory and related areas. In particular we mentioned some applications of each type of embedding.

Metric Embeddings into ℓ_p Spaces. ℓ_p spaces possess a natural geometric structure, especially ℓ_2 the Euclidean space, which has an inner product. This special structure is very helpful for solving various problems, even more so when the dimension is low. More interestingly, embeddings into ℓ_1 have implication for graph partitioning problems. Specifically, the ratio between the Sparsest Cut and the maximum multicommodity flow (called flow cut gap) is bounded by the distortion of the optimal embedding into ℓ_1 (see [LLR95, GNRS04]). In particular, if one embeds a graph into ℓ_1 with distortion t, it will imply a t-approximation to the sparsest cut problem.

We will be interested in finite subsets of ℓ_p spaces for $p \in [1, \infty]$. Every finite subset of ℓ_2 embeds isometrically to every ℓ_p for $p \in [1, \infty]$. Every finite metric space (even not ℓ_p) embeds isometrically into ℓ_∞. Every finite subset of ℓ_p space for $p \in [1, 2]$ embeds isometrically into ℓ_1. For any other pair ℓ_p, ℓ_q, there is no embedding with constant distortion for all finite subsets. See [Mat02] for details.

In a celebrated result, Bourgain [Bou85] showed that any metric space on n points embeds with distortion $O(\log n)$ into Euclidean space (and therefore to any ℓ_p). Linial, London, and Rabinovich [LLR95] have shown this to be tight.

If the source space X is n points in ℓ_2, a famous dimension reduction lemma by Johnson and Lindensstrauss [JL84], asserts that for every parameter $\epsilon \in (0, 1)$ X can be embedded into $\ell_2^{O(\log n/\epsilon^2)}$ (i.e., Euclidean space of dimension $O(\log n/\epsilon^2)$) with distortion $1+\epsilon$. This is an extremely useful lemma with applications for streaming algorithms, nearest neighbor search, compressed sensing and many more.

Metric Embeddings of Special Graph Families. Since general n-point metrics require $\Omega(\log n/p)$-distortion to embed into ℓ_p-norms, much attention was given to embeddings of restricted graph families that arise in practice. As the class of graphs embeddable with some distortion into some target normed space is closed under taking minors, it is natural to focus on minor-closed graph families. A long-standing conjecture in this area is that all non-trivial minor-closed families of graphs embed into ℓ_1 with distortion depending only on the graph family and not the size n of the graph.

Stochastic Metric Embeddings. Given a graph family $\mathcal{F}$, a *stochastic embedding* of $G=(V,E,w)$ into $\mathcal{F}$ is a distribution $\mathcal{D}$ over pairs (H,f_H) where $H\in\mathcal{F}$ and f_H is embedding of G into H. We say that $\mathcal{D}$ is dominating if for every $(H,f_H)\in\text{supp}(\mathcal{D})$, f_H is dominating. We say that a dominating[1] stochastic embedding $\mathcal{D}$ has expected distortion t, if for every pair $u,v\in V$ it holds that

$$\mathbb{E}_{(H,f_H)\sim\mathcal{D}}\left[d_H\left(f_H(u),f_H(v)\right)\right]\le t\cdot d_G(u,v)\ .$$

In a highly influential series of works by Bartal and Fakcharoenphol, Rao and Talwar [Bar96b, FRT04], it was shown that every n-point metric space has a stochastic embedding to the families of ultrametrics (or trees) with expected distortion $O(\log n)$ (which is also tight [Bar96b]). In some applications, it is important that the sampled tree will be a spanning tree rather than only dominating (e.g. for routing). In this case Abraham and Neiman [AN12] (following [EEST05]) showed an $\tilde{O}(\log n)$ expected distortion. Stochastic embeddings into trees have become a very basic technique in approximation and online algorithms, as trees are easy to work with and generally enjoy efficient algorithms.

Metric Data Structures. In some cases we might prefer to represent distances in a data structure rather than as a metric space / graph. Such a representation is often more computationally efficient, and might have better distortion. A *distance oracle* is a data structure that supports distance queries between vertex pairs. In the study of distance oracles, we look for tradeoffs between space, query time and distortion (the accuracy of the answers). Given an n-vertex graph and parameter $t=1,2,\dots$, Thorup and Zwick [TZ01a] constructed a distance oracle of size $O(t\cdot n^{1+1/t})$, $O(t)$ query time and distortion $2t-1$. In a recent series of works [WN13, Che14, Che15] the space and query time were improved to $O(n^{1+1/t})$ and $O(1)$ respectively.

An another example of metric data structure is a *distance labeling* [Pel99, GPPR01]. Here we assign each vertex a label, and identify a global function that, given two labels, can estimate the distance between the respective vertices. The goal is to optimize the tradeoff between the label size and distortion. The distance oracle of Thorup and Zwick [TZ01a] can be converted into a distance-labeling scheme with label size $O(n^{1/t}\cdot\log^{1-1/t}n)$ and distortion $(2t-1)$. An interesting special case is when the input graph is planar. Here an $1+\epsilon$ stretch labeling scheme is possible with label size $O(\frac{1}{\epsilon}\log n)$ [Tho01, Kle02].

A *routing scheme* in a network is a mechanism that allows packets to be delivered from any node to any other node. The network is represented as a weighted undirected graph, and each node can forward incoming data by using local information stored at the node, often called a routing table, and the (short) packet's header. The routing scheme has two main phases: in the preprocessing phase, each node is assigned a routing table and a short label. In the routing phase, each node receiving a packet should make a local decision, based on its own routing table and the packet's header (which may contain the label of the destination, or

[1] Recall that embedding $f_H:G\to H$ is dominating if there are no contractions.

a part of it), where to send the packet. The *routing decision time* is the time required for a node to make this local decision. The *stretch* of a routing scheme is the worst ratio between the length of a path on which a packet is routed, to the shortest possible path. The classical routing scheme of [TZ01b], for a parameter $k > 1$, provides a scheme with routing tables of size $O(k \cdot n^{1/k})$, labels of size $(1+o(1))k \log n$, stretch $4k-5$, and decision time $O(1)$ (but the initial decision time is $O(k)$). The stretch was improved recently to roughly $3.68k$ by [Che13].

Spanners. Given a n-vertex graph $G = (V, E, w)$ and a parameter $t \geq 1$, a subgraph $H = (V, E', w)$ of G ($E' \subseteq E$) is called a *t-spanner* for G if for all $u, v \in V$, $\delta_H(u, v) \leq t \cdot \delta_G(u, v)$. The parameter t is called the *stretch* of H. While minimizing the stretch we also wish the spanner to have a small number of edges. In addition, its *weight* $w(H) = \sum_{e \in E'} w(e)$ should be close to the weight of a minimum spanning tree (MST) of the graph G. The normalized notion of weight $\Psi(H) = \frac{w(H)}{w(MST(G))}$, is called *lightness*. Light and sparse spanners are particularly useful for broadcast protocols, network synchronization, data gathering, routing, sensor networks, VLSI circuit design and much more (see [FS16] for references and further applications).

The *greedy spanner*[2] by Althöfer et al. [ADD+93] is arguably the simplest and most well-studied spanner construction. Althöfer et al. [ADD+93], for every parameter $k \geq 1$, showed that the *greedy* $(2k-1)$-spanner has $O(n^{1+1/k})$ edges. Chandra et al. [CDNS92] proved that the greedy spanner with stretch parameter $t = (2k-1) \cdot (1+\epsilon)$ has lightness $O_\epsilon(k \cdot n^{1/k})$ [3]. Later, Elkin, Neiman, and Solomon [ENS14] improved the analysis of [CDNS92] and showed $O_\epsilon(\frac{k}{logk} \cdot n^{1/k})$ lightness. In a recent breakthrough, Chechik and Wulff-Nilsen [CW18] used a much more complicated algorithm and constructed an $(2k-1) \cdot (1+\epsilon)$ spanner with $O_\epsilon(n^{1/k})$ lightness. Under Erdős' girth conjecture [Erd64], the lightness is asymptotically tight up to the dependency on ϵ.

Das, Heffernan, and Narasimhan [DHN93] showed that in d dimensional Euclidean metrics[4], the greedy $(1+\epsilon)$-spanner has lightness $\epsilon^{-O(d)}$. For the case where the shortest path d_G of the input graph has doubling dimension[5] ddim, Gottlieb [Got15] constructed $1+\epsilon$ spanners with lightness $(\mathrm{ddim}/\epsilon)^{O(\mathrm{ddim})}$ (improving over [Smi09]).

Steiner Point Removal. In the *Steiner point removal* (SPR) problem we are given a subset of terminals $K \subseteq V$ of size k (the non-terminal vertices are called Steiner vertices). The goal is to construct a new graph $M = (K, E')$ with positive weight function w', with the terminals as its vertex set, such that: (1) M is a graph minor of G, and (2) the distance between every pair of terminals t, t' is distorted by at most a multiplicative factor of α (that is $\forall t, t' \in K, \quad d_G(t, t') \leq d_M(t, t') \leq \alpha \cdot d_G(t, t')$). Property (1) expresses preservation of the topological structure of the original graph. For example if G was planar, so will M be, whereas property (2) expresses preservation of the geometric structure of the original graph, that is, distances between terminals. The question is: what is the minimal α (which may depend on k) such that every graph with a terminal set of size k will admit a solution to the SPR problem with distortion α.

The underlying fundamental question is the following: given some graph family $\mathcal{F}$, is the collection of geometries obtained by k-vertex graphs from $\mathcal{F}$ can be significantly different from the collection of geometries obtained by restricting the attention to k terminals in a big graphs from $\mathcal{F}$?

[2]The greedy spanner H with parameter t is constructed by repeatedly adding an edge between the closest pair of neighboring vertices $\{u, v\}$ such that $d_H(u, v) > t \cdot d_G(u, v)$.

[3]In O_ϵ notation we hide polynomial factors in ϵ.

[4]By d dimensional Euclidean metric here we mean a complete graph on n vertices, where each vertex v is associated with a point $p_v \in \mathbb{R}_d$ such that the weight of the edge $\{u, v\}$ equals $\|p_v - p_u\|_2$.

[5]The *doubling dimension* of a metric space (M, δ) is the smallest value ddim such that every ball B in the metric space can be covered by at most 2^{ddim} balls of half the radius of B.

The minor restriction ensures that the graph on the terminals will remain in the family. However it has additional advantages. Suppose that the given graph is planar and all the terminals lie on a single face (Okamura-Seymour instance), then every minor restricted to the terminals will be an outerplanar.

If the given graph G is a tree, Gupta [Gup01] constructed a minor with distortion 8, which is tight by Chan et al. [CXKR06]. This lower bound of 8 is the best known lower bound for general graphs as well. Basu and Gupta [BG08] showed that on outerplanar graphs, the SPR problem can be solved with distortion $O(1)$. Kamma, Krauthgamer, and Nguyen [KKN15] provided an $O(\log^5 k)$ upper bound for general graphs, which was recently improved to $O(\log^2 k)$ by Cheung [Che18].

Englert et al. [EGK+14] showed that every graph G, admits a distribution $\mathcal{D}$ over terminal minors with expected distortion $O(\log k)$. Further, if the graph is β-decomposable, it admits a distribution with $O(\beta \log \beta)$ expected distortion. In particular, planar graphs and graphs excluding a fixed minor are $O(1)$-decomposable.

Krauthgamer, Nguyen, and Zondiner [KNZ14] showed that if we allow the minor M to contain at most $O(k^4)$ Steiner vertices (in addition to the terminals), then distortion 1 can be achieved. They further showed that for graphs with constant treewidth, $O(k^2)$ Steiner points will suffice for distortion 1. Cheung, Gramoz, and Henzinger [CGH16] showed that allowing $O(k^{2+\frac{2}{t}})$ Steiner vertices, one can achieve distortion $2t-1$ (in particular distortion $O(\log k)$ with $O(k^2)$ Steiners). For planar graphs, [CGH16] achieved $1+\epsilon$ distortion with $\tilde{O}((\frac{k}{\epsilon})^2)$ Steiner points.

Sparsifiers. In metric embeddings, spanners, etc., we look for succinct representation of graphs while preserving the geometry, i.e., distances between vertices. However, there are other graph properties that one might wish to preserve while using a succinct representation. The seminal work of Benczúr and Karger [BK96] showed that every edge-weighted undirected graph admits *cut-sparsification* within factor $(1+\epsilon)$ using $O(\epsilon^{-2} n \log n)$ edges. More precisely, let $\mathsf{Cut}_G(S)$ denote the total weight of edges in G that have exactly one endpoint in S. Then for every such G and $\epsilon \in (0,1)$, there is a re-weighted subgraph $G_\epsilon = (V, E_\epsilon \subseteq E, w_\epsilon)$ with $|E_\epsilon| \le O(\epsilon^{-2} n \log n)$ edges, such that

$$\forall S \subset V, \qquad \mathsf{Cut}_{G_\epsilon}(S) \in (1 \pm \epsilon) \cdot \mathsf{Cut}_G(S), \tag{1}$$

and moreover, such G_ϵ can be computed efficiently.

This sparsification methodology turned out to be very influential. The original motivation was to speed up algorithms for cut problems – one can compute a cut sparsifier of the input graph and then solve an optimization problem on the sparsifier – and indeed this has been a tremendously effective approach, For example, see [BK96, BK02, KL02, She09, Mad10]. Another application of this remarkable notion is to reduce space requirement, either when storing the graph or in streaming algorithms [AG09]. In fact, followup work offered several refinements, improvements, and extensions (such as to spectral sparsification), see [ST04b, ST11, SS11, dCHS11, FHHP11, KP12, NR13, BSS14, KK15]. The current bound for cut sparsification is $O(n/\epsilon^2)$ edges, proved by Batson, Spielman and Srivastava [BSS14], and it is known to be tight [ACK+16].

1.1 Refined Notions of Embeddings

Consider a non-contractive embedding $\phi : X \to Y$. The distortion of the pair x, y is $\frac{d_Y(\phi(x),\phi(y))}{d_Y(x,y)}$. Thus the distortion of the embedding ϕ is simply the worst case (maximal) distortion over all the pairs. This is the definition all previous results coped with. A natural disadvantage of these results is the dependence of all the relevant parameters on n, the cardinality of the input graph/metric. Nevertheless, most of these results are either completely tight, or very close to being so. Several approaches to cope with this shortcoming were proposed.

Terminal Embeddings. Here we are given a set $K \subseteq X$ of points of size k, which are designated as *terminals*. The objective is to embed the metric into a simpler metric, while approximately preserving the distances between the terminals to *all other points*. Formally, the terminal distortion of an embedding $\phi : X \to Y$ is the maximal distortion over all pairs in $K \times X$. Terminal embeddings have implications for the areas of approximation and online algorithms.

This notion of distortion was studied in the master thesis of the author, and in particular published in [EFN17] co-authored with Elkin and Neiman. In many cases, the cardinality of the input metric n can be replaced by that of the terminal set k. Some notable results are as follows: embedding of a general metric into ℓ_2 with terminal distortion $O(\log k)$, spanner construction with terminal distortion $4t-1$ and $O(n+\sqrt{n}\cdot k^{1+\frac{1}{t}})$ edges, construction of a single spanning tree with terminal distortion $2k-1+\epsilon$ and lightness $O(\frac{k}{\epsilon})$ for any $\epsilon > 0$, stochastic embedding into spanning trees with expected terminal distortion $\tilde{O}(\log k)$, and more.

In a follow up paper, Elkin and Neiman [EN18] study terminal embedding of metric spaces with constant doubling metrics. In particular they constructed a spanner with $1+\epsilon$ terminal distortion and $n+o(n)$ edges. Additionally, they constructed a labeling scheme with $\approx \log k$ label size.

Recently, Mahabadi et al. [MMMR18] answered a question from our paper [EFN17], showing a terminal version of the JL lemma. Specifically, they show that given a set K of k points in $\mathbb{R}^d$, it is possible to embed all of $\mathbb{R}^d$ into $\mathbb{R}^{\log k/\epsilon^4}$ with terminal distortion $1+\epsilon$. Even more recently Narayanan and Nelson [NN18] were able to reduce the number of dimensions to $\frac{\log k}{\epsilon^2}$, which is also tight. These new embeddings are called *Terminal JL*.

Scaling Distortion. Another approach to cope with large worst-case distortion bounds is to construct embeddings where some pairs of vertices/points enjoy better guarantees. Specifically, [KSW04, ABC+05, ABN11, CDG06] studied embeddings in which the distortion of at least $1-\epsilon$ fraction of the pairs is improved as a function of ϵ, for all $\epsilon \in [0,1]$ simultaneously. Formally, given a function $\alpha : (0,1) \to \mathcal{R}_+$ we say that embedding $\phi : X \to Y$ has scaling distortion α if for every $\epsilon \in (0,1)$ at most ϵ fraction of the pairs (that is $\epsilon \cdot \binom{|X|}{2}$) suffer from distortion $\alpha(\epsilon)$ or larger. Some notable results being: an embedding of a general metric space into ℓ_2 with scaling distortion $O(\log\frac{1}{\epsilon})$, and stochastic embedding into trees with expected scaling distortion $O(\log\frac{1}{\epsilon})$. Note that while the worst case distortion is $O(\log n)$ (fixing $\epsilon = \frac{1}{n^2}$), half of the pairs are guaranteed constant distortion! A nice property of scaling distortion is that it is also provides constant average distortion[6] $\sum_{x,y\in\binom{X}{2}} \frac{d_Y(\phi(x),\phi(y)}{d_X(x,y)} = O(1)$ [7].

1.2 Related Work

While the minor embedding conjecture of [GNRS04] remains unresolved in general, some progress has been made on special classes of graphs. The class of outerplanar graphs (which exclude $K_{2,3}$ and K_4 as a minor) embeds isometrically into ℓ_1; this follows from results of Okamura and Seymour [OS81] as was proved by Hurkens, Schrijver, and Tardos [HST86]. Following [GNRS04], Chakrabarti et al. [CJLV08] show that every graph with treewidth-2 (which excludes K_4 as a minor) embeds into ℓ_1 with distortion 2 (which is tight, as shown by [LR10]). Lee and Sidiropoulos [LS13] showed that every graph with pathwidth k can be embedded into ℓ_1 with distortion $(4k)^{k^3+1}$. Chekuri et al. [CGN+03] extend the Okamura and Seymour bound for outerplanar graphs to k-outerplanar graphs, and showed that these embed into ℓ_1 with distortion $2^{O(k)}$. Rao [Rao99] (see also [KLMN04]) embed planar graphs into ℓ_p with distortion $O(\log^{1/p} n)$. For

[6] As long as the scaling distortion is smaller than $O(\epsilon^{-\delta})$ for some $\delta < 1$.

[7] There are alternative definitions of average distortion that could be found in the literature, see related works.

graphs with genus g, [LS10] showed an embedding into Euclidean space with distortion $O(\log g + \sqrt{\log n})$. Finally, for H-minor-free graphs, combining the results of [AGG+14, KLMN04] gives ℓ_p-embeddings with $O(|H|^{1-1/p} \log^{1/p} n)$ distortion.

Some progress has also been made on stochastic embeddings. Gupta et al. [GNRS04] showed that outerplanar graphs embed into trees with $O(1)$ expected distortion. Lee and Sidiropoulos [LS13] showed that pathwidth k graphs embedded into trees with distortion $(4k)^{k^3+1}$. On the negative side, [LS13] showed that pathwidth $k+1$ graphs cannot be stochastically embedded into pathwidth k graphs with constant distortion. Further, Gupta et al. [GNRS04] showed that already planar graphs (or even treewidth 2 graphs) cannot be embedded into trees with any constant distortion.

Rabinovich [Rab08] defined the average distortion of a dominating embedding $f : X \to Y$ as $\frac{\sum_{x,y\in X} d_Y(f(x),f(y))}{\sum_{x,y\in X} d_X(x,y)}$.

2 Results

In the introduction section we presented the state of the art in various metric embeddings related topics, as it was before the contribution in this thesis, as well as other follow-up contributions. In Section 2.1 we describe the results presented in this thesis. Afterwards, in Section 2.2 we describe our results that were not fortunate enough to make it into the thesis. We would like to emphasize that the criteria for entering the thesis was rather technical than qualitative (journal publication).

2.1 Results Presented in this Thesis

The descriptions of the results is organized according to the papers constituting the thesis.

Prioritized Embedding ([EFN15, EFN18]). An inherent shortcoming of scaling distortion is that the pairs that enjoy better than worst-case distortion cannot be specified in advance. We introduce a novel definition of distortion called priority distortion. Here, in addition to the metric space (X, d), we are given an ordering of the metric points $X = (x_1, \dots, x_n)$ *arbitrarily in advance*, and devise an embedding in which the distortion of the pair $\{x_i, x_j\}$ depends on $\min\{i, j\}$, regardless of the cardinality of the metric space. In many cases, we are able to construct embeddings such that the guarantee for low priority pairs is similar to the worst case guarantee in the classic setting, while the guarantee for high priority pairs are considerably improved. Hence our results are stronger.

Formally, for a function $\alpha : \mathbb{N} \to \mathbb{R}_+$, we say that embedding ϕ has prioritized distortion α if for all $1 \le j < i \le n$,

$$d_X(x_j, x_i) \le d_Y(\phi(x_j), \phi(x_i)) \le \alpha(j) \cdot d_X(x_j, x_i) \ .$$

Partial list of results:

- Embedding into ℓ_p. Every metric space embeds into ℓ_p space with prioritized distortion $\tilde{O}(\log j)$. By [LLR95], an $\Omega(\log j)$ lower bound on prioritized distortion follows. Thus the result is tight up to second order factors. We close this gap in the paper presented next [BFN19].
- Embedding into ℓ_p with prioritized dimension. We say that point x has prioritized dimension β, if for every $j \in [n]$, only the first $\beta(j)$ coordinates in $\phi(x_j)$ may be nonzero. We showed that every metric space embeds into ℓ_p space with prioritized distortion[8] polylog(j) and prioritized dimension polylog(j).

[8]Actually in this case the distortion guarantee is changed to $\frac{1}{\alpha(j)} \cdot d_X(x_j, x_i) \le d_Y(\phi(x_j), \phi(x_i)) \le d_X(x_j, x_i)$.

- Stochastic Embedding. Every metric space admits a stochastic embedding into trees with expected prioritized distortion $O(\log j)$. This result is also tight [Bar96a].

- Embedding into a single tree. Define Φ to be the family of non-decreasing functions $\alpha : \mathbb{N} \to \mathbb{R}_+$ such that $\sum_{i=1}^{\infty} 1/\alpha(i) \leq 1$. Then for any finite metric space (X, d) and any $\alpha \in \Phi$, there is a (non-contractive) embedding of X into a single tree with priority distortion $2\alpha(j)$. This result is tight (up to a constant). That is, if $\sum_{i=1}^{\infty} 1/\alpha(i) > 1$ then embedding into a single tree with prioritized distortion α is impossible. As an example, this result implies that every metric embeds into a single tree with prioritized distortion $\tilde{O}(j)$.

- Distance Oracle & Labeling. For distance oracles we got several tradeoffs between space and prioritized distortion. For distance labeling, we offer a construction where in addition to prioritized distortion, we have prioritized label size. This could be useful in a setting where the high ranked points participate in numerous computations, as representing these points requires very few coordinates. We can thus store many of them in the cache or other high speed memory. All the tradeoffs are presented in the table below.

Distance Oracle		
Priority Distortion	Space	Query time
$O\left(\frac{\log n}{1+\log(n/j)}\right)$	$O(n \log\log\log n)$	$O(1)$
$2\lceil\frac{t\log j}{\log n}\rceil - 1$	$O(tn^{1+1/t})$	$O(\lceil\frac{t\log j}{\log n}\rceil)$
$2\log j - 1$	$O(n\log n)$	$O(\log j)$
Labeling Scheme		
Priority Distortion	Prioritized label size	
$2\lceil\frac{t\log j}{\log n}\rceil - 1$	$O(n^{1/t} \cdot \log j)$	
$2\log j - 1$	$O(\log j)$	
$1+\epsilon$	$O(\frac{1}{\epsilon}\log j)$ (for graphs excluding a fixed minor)	

- Routing Scheme. Given a priority ranking and a parameter $t \geq 1$, we construct a routing scheme, such that the label size of x_j is at most $\log j \cdot \lceil\frac{t\log j}{\log n}\rceil \cdot (1+o(1))$, its header of size $\log j \cdot (1+o(1))$, and it stores a routing table of size $O(n^{1/t} \cdot \log j)$. Routing from any vertex into x_j will have stretch at most $4\lceil\frac{t\log j}{\log n}\rceil - 3$. In particular, for $t = \log n$ we roughly have labels of size $\log^2 j$, header $\log j$, routing table $O(\log j)$ and stretch $O(\log j)$.

On Notions of Distortion and an Almost Minimum Spanning Tree with Constant Average Distortion [BFN16, BFN19]. In scaling distortion [ABN11] we are guaranteed that most of the pairs will suffer from small distortion only. In particular, scaling distortion implies constant average distortion. However, there is no way to choose which pairs will enjoy small distortion. On the other hand, in priority distortion [EFN18] we may choose the priority, and can guarantee small distortion for points of high importance. However, most of the pairs might suffer from high distortion. In particular, the average distortion might be almost as large as the worst case.

At first glance, these two notions of distortion seem very different. The most surprising ingredient of this work is a *general reduction* relating the notions of prioritized distortion and scaling distortion. In fact, we show that prioritized distortion is essentially equivalent to a strong version of scaling distortion called *coarse* scaling distortion, in which for every point p and every $0 < \epsilon < 1$, the distances to the $1-\epsilon$ fraction of the farthest points from p are preserved with the desired distortion. We prove that there is a particular priority π such that any embedding with a prioritized distortion α (w.r.t. π) has coarse scaling distortion bounded by $O(\alpha(8/\epsilon))$. We further show a reduction in the opposite direction, informally, that given an embedding with coarse scaling distortion γ, there exists an embedding with prioritized distortion $\gamma(\mu(j))$, where μ is a function such that $\sum_i \mu(i) = 1$ (e.g., $\mu(j) = \tilde{\Theta}(\frac{1}{j})$). We note that this reduction heavily relies on the property of coarse scaling distortion embeddings and does not apply to non-coarse scaling embeddings. Yet, most existing scaling embeddings are indeed coarse. This result implies that all existing priority distortion results have their coarse scaling distortion counterparts, and vice versa. In particular, this equivalence implies many new results on refined notions of distortion. See the list below.

A less direct application of the equivalence theorem is a construction that, given a weighted graph, provides a spanning tree whose weight is at most $(1+\rho)$ times that of the MST, while having $O(1/\rho)$ average distortion. We show this tradeoff to be tight. This result may be of interest for network applications. It is extremely common in the area of distributed computing that an MST is used for communication between the network nodes. This allows easy centralization of computing processes and an efficient way of broadcasting through the network, allowing communication to all nodes at a minimum cost. Yet, when communication is required, the cost of routing through the MST may be extremely high, even between nearby points. However, in practice it is the average distortion, rather than the worst-case distortion, that is often used as a practical measure of quality, as has been a major motivation behind the initial work of [KSW04, ABN11, ABN15]. The MST still fails even in this relaxed measure. Our result overcomes this by promising small routing cost between nodes on average, while still possessing the low cost of broadcasting through the tree, thereby maintaining the standard advantages of the MST.

A partial list of new results on refined notions of distortion proved in this paper appears below. The only one proven directly is the spanner with lightness $1+\rho$ and prioritized distortion $\tilde{O}(\log j)/\rho$. All others follow from the equivalence theorem.

- Embedding into ℓ_p: For $p \in (0,1)$ every metric space embeds into ℓ_p space with prioritized distortion $O(\log j)$ (removing the log log factors from [EFN18]).
- Decomposable Metric: For $p \in (0,1)$ every τ-decomposable metric space embeds into ℓ_p with prioritized distortion $O(\tau^{1-1/p}(\log j)^{1/p})$.
- Distance Oracle: For every metric space there exists a distance oracle with $O(n)$ space, $O(1)$ query time and $O(\log j)$ priority distortion (improving over [EFN18]).
- Graph spanners: Given a weighted graph G there is a:
  - Spanner with $O(n)$ edges and $O(\log j)$ prioritized distortion.
  - Spanner with lightness $1+\rho$ and prioritized distortion $\tilde{O}(\log j)/\rho$, for arbitrarily small parameter $\rho \in (0,1)$.
  - Spanner with lightness $1+\rho$ and coarse scaling distortion $\tilde{O}(\log 1/\epsilon)/\rho$, for arbitrarily small parameter $\rho \in (0,1)$.
  - Spanning tree with lightness $1+\rho$ and scaling distortion $\tilde{O}(\sqrt{1/\epsilon})/\rho$, for arbitrarily small parameter $\rho \in (0,1)$.

Steiner Point Removal with distortion $O(\log k)$, using the `Relaxed-Voronoi` algorithm ([Fil18, Fil19]). In this paper we study the SPR problem on general graphs. The previous works [KKN15, Che18] constructed minors using the `Ball-growing` algorithm. In this paper we devise a novel algorithm called the `Relaxed-Voronoi` algorithm. The main contribution of this paper is a new upper bound of $O(\log k)$ for the

SPR problem. Furthermore, the `Relaxed-Voronoi` algorithm is simpler and more intuitive compared to the `Ball-growing` algorithm. Both algorithms grow clusters around the terminals, the main difference is that the `Ball-growing` algorithm has many iterations, growing slowly from all terminals (almost in parallel), while the `Relaxed-Voronoi` algorithm has one round only (each terminal construct a cluster by turn and done).

Additionally, we devise an efficient implementation of the `Relaxed-Voronoi` algorithm in almost linear time $O(m + \min\{m, nk\} \cdot \log n)$ $(m = |E|)$. While the `Ball-growing` algorithm can be implemented in polynomial time, it is not clear how to do so efficiently.

Sparsification of Two-Variable Valued CSPs ([FK17]). A valued constraint satisfaction problem (VCSP) instance (V, Π, w), is a set of variables V, with a set of constraints Π weighted by w. The value of an assignment of values to the variables is the total weight of the satisfied constrains. Following cut sparsification, we study the analogous problem of sparsifying VCSP, which was raised in [KK15, Section 4]. Given a VCSP instance, we are interested in a re-weighted sub-instance $(V, \Pi' \subset \Pi, w')$ that preserves the value of the given instance (under every assignment to the variables) within factor $1 \pm \epsilon$. Such sparsification of CSPs can be used to reduce storage space and running time of many algorithms.

We restrict our attention to two-variable constraints (i.e., of arity 2) over boolean domain (i.e., alphabet of size 2). To simplify matters even further, we focus on the case where all the constraints use the same predicate $P : \{0,1\}^2 \to \{0,1\}$. This restricted case of VCSP sparsification already generalizes cut-sparsification — simply representing every vertex $v \in V$ by a variable x_v, and every edge $(v, u) \in E$ by the constraint $x_v \neq x_u$. Observe that such VCSPs capture also other interesting graph problems, such as the *uncut edges* (using the predicate $x_v = x_u$), *covered edges* (using the predicate $x_v \vee x_u$) or the *directed-cut edges* (using the predicate $x_v \wedge \neg x_u$).

For CSPs consisting of a single predicate $P : \{0,1\}^2 \to \{0,1\}$, we show that a $(1+\epsilon)$-sparsifier of size $O(n/\epsilon^2)$ always exists if and only if $|P^{-1}(1)| \neq 1$ (i.e., P has 0,2,3 or 4 satisfying inputs). Observe that the latter condition includes the two graphical examples above of uncut edges and covered edges, but excludes directed-cut edges. We further show that our sparsity bound above is tight, except for some relatively trivial predicates P. We then build on our sparsification result to obtain $(1+\epsilon)$-sparsifiers for other CSPs, including 2SAT (which uses 4 predicate types) and 2LIN (which uses 2 predicate types).

In a recent follow-up, Butti and Živný [BZ19] generalize our result for binary predicates to any finite domain (as oppose to our $\{0,1\}$). They show that a predicate $P : D^2 \to \{0,1\}$ admits a sparsifier if and only if there are no $A, B \subset D$ of size 2 such that P restricted to $A \times B$ has a single 1 in its truth table.

2.2 Results: Related, Published During the PhD, but do not appear in the Thesis

The Greedy Spanner is Existentially Optimal ([FS16]). The greedy spanner is arguably the simplest and most well-studied spanner construction. Experimental results demonstrate that it is at least as good as any other spanner construction, in terms of both the size and weight parameters. However, a rigorous proof for this statement has remained elusive.

In this work we fill in the theoretical gap via a surprisingly simple observation: The greedy spanner is *existentially optimal* (or existentially near-optimal) for several important graph families, in terms of both the size and weight. Roughly speaking, the greedy spanner is said to be existentially optimal (or near-optimal) for a graph family $\mathcal{G}$ if the worst performance of the greedy spanner over all graphs in $\mathcal{G}$ is just as good (or nearly as good) as the worst performance of an optimal spanner over all graphs in $\mathcal{G}$.

Focusing on the weight parameter, the state-of-the-art spanner constructions for both general graphs [CW18] and doubling metrics [Got15] are complex. Plugging our observation into these results, we conclude that the greedy spanner achieves near-optimal weight guarantees for both general graphs and doubling metrics, thus resolving two longstanding conjectures in the area.

Further, we observe that approximate-greedy spanners are existentially near-optimal as well. Consequently, we provide an $O(n \log n)$-time construction of $(1+\epsilon)$-spanners for doubling metrics with constant lightness and degree. Our construction improves Gottlieb's [Got15] construction, whose runtime is $O(n \log^2 n)$ and whose number of edges and degree are unbounded, and remarkably, it matches the state-of-the-art Euclidean result (due to Gudmundsson et al. [GLN02]) in all the involved parameters (up to dependencies on ϵ and the dimension).

Light Spanners for High Dimensional Norms via Stochastic Decompositions ([FN18]). Spanners for low dimensional spaces (e.g., Euclidean space of constant dimension, or doubling metrics) are well understood. This lies in contrast to the situation in high dimensional spaces, where except for the work of Har-Peled, Indyk and Sidiropoulos [HPIS13], who showed that any n-point Euclidean metric has an $O(t)$-spanner with $\tilde{O}(n^{1+1/t^2})$ edges, little is known.

In this paper we study several aspects of spanners in high dimensional normed spaces. First, we build spanners for finite subsets of ℓ_p with $1 < p \leq 2$. Second, our construction yields a spanner which is both sparse and light. In particular, we show that any n-point subset of ℓ_p for $1 < p \leq 2$ has an $O(t)$-spanner with $n^{1+\tilde{O}(1/t^p)}$ edges and lightness $n^{\tilde{O}(1/t^p)}$.

Our results can also be applied more generally to any metric space admitting a certain low diameter stochastic decomposition. It is known that arbitrary metric spaces have an $O(t)$-spanner with lightness $O(n^{1/t})$. We exhibit the following tradeoff: metrics with decomposability parameter $\nu = \nu(t)$ admit an $O(t)$-spanner with lightness $\tilde{O}(\nu^{1/t})$. For example, metrics with doubling constant λ, graphs of genus g, and graphs of treewidth k, all have spanners with stretch $O(t)$ and lightness $\tilde{O}(\lambda^{1/t})$ (resp. $\tilde{O}(g^{1/t})$, $\tilde{O}(k^{1/t})$). While these families do admit a $(1+\epsilon)$-spanner, its lightness depends exponentially on the dimension (resp. $\log g$, $\log k$). Our construction alleviates this exponential dependency, at the cost of incurring larger stretch.

Constructing Light Spanners Deterministically in Near-Linear Time ([ADF+19]). In their recent breakthrough, Chechik and Wulff-Nilsen [CW18] improved the lightness of the state-of-the-art $(2k-1)(1+\epsilon)$-spanner construction to $O_\epsilon(n^{1/k})$ lightness. Soon after, the author and Solomon [FS16] showed that the classic greedy spanner construction achieves the same bounds. The major drawback of the greedy spanner is its running time of $O(mn^{1+1/k})$ (which is faster than [CW18]). This makes the construction impractical even for graphs of moderate size. Much faster spanner constructions do exist but they only achieve lightness $\Omega_\epsilon(kn^{1/k})$, even when randomization is used.

The contribution of this paper is fast deterministic spanner constructions, and achieve similar bounds as the state-of-the-art slower constructions. Our first result is an $O_\epsilon(n^{2+1/k+\epsilon'})$ time spanner construction which achieves the state-of-the-art bounds. Our second result is an $O_\epsilon(m + n \log n)$ time construction of a spanner with $(2k-1)(1+\epsilon)$ stretch, $O(\log k \cdot n^{1+1/k})$ edges and $O_\epsilon(\log k \cdot n^{1/k})$ lightness. For the case $k = \log n$ this is an exponential improvement in the dependence on k compared to the previous result with such running time. Finally, for the important special case where $k = \log n$, for every constant $\epsilon > 0$, we provide an $O(m + n^{1+\epsilon})$ time construction that produces an $O(\log n)$-spanner with $O(n)$ edges and $O(1)$ lightness which is asymptotically optimal. This is the first known sub-quadratic construction of such a spanner for any $k = \omega(1)$. We describe our results and compare them to previous ones in the table below.

Stretch	Size	Lightness	Construction	Ref
$(2k-1)(1+\epsilon)$	$O\left(n^{1+1/k}\right)$	$O\left(n^{1/k}\right)$	$n^{\Theta(1)}$	[CW18]
$(2k-1)(1+\epsilon)$	$O\left(n^{1+1/k}\right)$	$O\left(n^{1/k}\right)$	$O\left(mn^{1+1/k}\right)$	[FS16]
$(2k-1)$	$O\left(kn^{1+1/k}\right)$	no bound	$O\left(km\right)$	[BS07, RTZ05]
$(2k-1)(1+\epsilon)$	$O\left(kn^{1+1/k}\right)$	$O\left(kn^{1/k}\right)$	$O\left(km+n\log n\right)$	[ES16]
$O(k)$	$O(\log k\cdot n^{1+1/k})$	no bound	$O(m+n\cdot\log k)$	[MPVX15]
$(2k-1)(1+\epsilon)$	$O(\log k\cdot n^{1+1/k})$	$O\left(k\cdot n^{1/k}\right)$	$O(m+n\cdot\log n)$	[EN17]
$(2k-1)(1+\epsilon)$	$O\left(\log k\cdot n^{1+1/k}\right)$	$O\left(\log k\cdot n^{1/k}\right)$	$O(m+n\cdot\log n)$	[ADF+19]
$(2k-1)(1+\epsilon)$	$O\left(n^{1+1/k}\right)$	$O\left(n^{1/k}\right)$	$O(n^{2+1/k+\epsilon'})$	[ADF+19]
$O(k)$	$O\left(n^{1+1/k}\right)$	$O\left(n^{1/k}\right)$	$O\left(m+n^{1+\epsilon'+1/k}\right)$	[ADF+19]
$O(\log n)/\delta$	$O\left(n\right)$	$1+\delta$	$O\left(m+n^{1+\epsilon'}\right)$	[ADF+19]

To achieve our constructions, we show a novel deterministic incremental approximate distance oracle. Our new oracle is crucial in our construction, as known randomized dynamic oracles require the assumption of a non-adaptive adversary. This is a strong assumption, which has seen recent attention in prolific venues. Our new oracle allows the order of the edge insertions to not be fixed in advance, which is critical as our spanner algorithm chooses which edges to insert based on the answers to distance queries. We believe our new oracle is of independent interest.

Ramsey Spanning Trees and their Applications ([ACE+18]). The *metric Ramsey problem* asks for the largest subset S of a metric space that can be embedded into an ultrametric (more generally into ℓ_2) with a given distortion. Study of this problem was motivated as a non-linear version of Dvoretzky theorem. Mendel and Naor [MN07] devised the so called Ramsey Partitions to address this problem, and showed the algorithmic applications of their techniques to approximate distance oracles and ranking problems.

In this paper we study the natural extension of the metric Ramsey problem to graphs, and introduce the notion of *Ramsey Spanning Trees*. We ask for the largest subset $S\subseteq V$ of a given graph $G=(V,E)$, such that there exists a spanning tree of G that has small stretch for S. Applied iteratively, this provides a small collection of spanning trees, such that each vertex has a tree providing low stretch paths to *all other vertices*. The union of these trees serves as a special type of spanner, a *tree-padding spanner*. We use this spanner to devise the first compact stateless routing scheme with $O(1)$ routing decision time, and labels which are much shorter than in all currently existing schemes.

We first revisit the metric Ramsey problem, and provide a new deterministic construction. We prove that for every k, any n-point metric space has a subset S of size at least $n^{1-1/k}$ which embeds into an ultrametric with distortion $8k$. We use this result to obtain the state-of-the-art deterministic construction of a distance oracle. Building on this result, we prove that for every k, any n-vertex graph $G=(V,E)$ has a subset S of size at least $n^{1-1/k}$, and a spanning tree of G, that has terminal distortion $O(k\log\log n)$ w.r.t. S.

Metric embedding via shortest path decompositions ([AFGN18]). In this paper we study embeddings of special graph families into ℓ_p spaces. We devise embeddings for any graph family which admits "shortest path decompositions" (SPD) of "low depth". Every (weighted) path graph has an SPD of depth 1. A graph G has an SPD of depth k if after removing some *shortest path* P, every connected component in $G\setminus P$ has an SPD of depth $k-1$. The main result of this paper is that every weighted graph with an SPD of depth k, is embeddable into ℓ_p with distortion $O(k^{\min\{1/p,1/2\}})$. This result is tight for every $p>1$.

We summarize the implications for various graph families in the table below.

Graph Family	Our results	Previous results	
Pathwidth k	$O(k^{1/p})$	$(4k)^{k^3+1}$ into ℓ_1	[LS13]
Treewidth k	$O((k\log n)^{1/p})$	$O(k^{1-1/p}\cdot\log^{1/p}n)$	[KLMN04]
		$O((\log(k\log n))^{1-1/p}(\log^{1/p}n))$	[KK16]
Planar	$O(\log^{1/p}n)$	$O(\log^{1/p}n)$	[Rao99]
H-minor-free	$O((g(H)\log n)^{1/p})$	$O(\|H\|^{1-1/p}\log^{1/p}n)$	[AGG+14]+[KLMN04]

For bounded pathwidth graphs we provide super-exponential improvement for the case $p = 1$, while having completely new results for every $p > 1$. For bounded treewidth graphs we improve the state of the art for the case where $p > 2$. For minor free graphs we provide improvement for large enough values of p. Finally, for planar graphs we just re-proved the celebrated result of Rao, while using completely different techniques.

Relaxed Voronoi: A Simple Framework for Terminal-Clustering Problems ([FKT19]). This is a follow-up paper to [Fil19]. We used the `Relaxed-Voronoi` framework presented there to reprove three known algorithmic bounds for terminal-clustering problems. In this genre of problems, the input is a metric space (X, d) (possibly arising from a graph) and a subset of terminals $K \subset X$, and the goal is to partition the points X such that each part, called a cluster, contains exactly one terminal (possibly with connectivity requirements) so as to minimize some objective. The three bounds we reprove are for Steiner Point Removal on trees [Gup01], for Metric 0-Extension in bounded doubling dimension [LN03], and for Connected Metric 0-Extension [EGK+14].

The `Relaxed-Voronoi` framework was already employed successfully to provide state-of-the-art results for terminal-clustering problems on general metrics [CKR01, Fil19]. However, for restricted families of metrics, e.g., trees and doubling metrics, only more complicated, ad-hoc algorithms are known. Our main contribution is to demonstrate that the `Relaxed-Voronoi` algorithm is applicable to restricted metrics, and actually leads to relatively simple algorithms and analyses.

3 Summary, Discussion and Open Problems

Classically, most of the results in metric embedding theory, and more generally in theoretical computer science, are concerned with analyzing the worst case scenario. One reason is that it is usually easier to rigorously analyze worst case, while it is much harder to give a more precise description of richer behaviors. This approach often gives overwhelming importance to outliers that essentially could be neglected. On the other hand, industry and more practically oriented fields of study, are interested in "better descriptions" of performance, and are not willing to be satisfied with worst case only. However, their analysis is typically based on experiments, while the algorithms are just heuristics. In other words, they sometimes lack a stable theoretical foundation. Understanding this phenomena and giving rigorous explanations is a fascinating theoretical question. Even more importantly, once a phenomenon is fully understood, we gain a much stronger advantage using it.

The most famous example is the Simplex algorithm for Linear programming. The Simplex algorithm has been used very successfully in the industry since the late 1940s. However, it was shown that in the worst case its running time is exponential. It took some time, an only in the early 1980s was a polynomial time algorithm for linear programming discovered. Nevertheless, the industry kept using the Simplex algorithm, as apparently in practice it is much more efficient. The Simplex algorithm lacked any theoretical explanation for its excellent performance. Finally, Spielman and Teng came up with a smooth analysis for the Simplex algorithm. They proved that the cases where the runtime of the Simplex algorithm is exponential are isolated and essentially negligible. More formally, they show that given a linear programming instance, if we add random small perturbations to the constraints, then w.h.p. the Simplex will run in only polynomial time.

The main theme of this thesis is the construction of metric embedding with refined guarantees. That is, our goal is to give rigorous theorems explaining a more subtle behavior than simply worst case. Indeed, we proved some theorems that cannot be described using the crude notion of worst case. We started by defining prioritized distortion. We constructed various embeddings with prioritized distortion, emphasizing the phenomenon that generally, the distortion could be a function of the relative ranking, rather than the same worst case for all points. Further, we study the previously introduced scaling distortion. Even though intuitively priority and (coarse) scaling distortion significantly differ, we prove that they are essentially equivalent. This equivalence theorem implies many new results on refined embeddings. Another interesting application is the construction of a tree with $1+\rho$ lightness and $O(1/\rho)$ average distortion for every $\rho \in (0,1)$.

Next we turn to study the fundamental question of Steiner point removal. Consider k terminals in some huge planar graph. Is there a planar graph supported only on these terminals that (approximately) preserves the distances between terminals? What is the best possible distortion? While we were not able to answer this question, we provide an $O(\log k)$ upper bound for general graphs (for SPR), which is also the best known for planar graphs and for the question above.

The best known lower bound for the SPR problem is 8 [CXKR06]. This bound is achieved using the unweighted full binary tree with the leaves being the terminals and depth tending to infinity. Once we analyze more complicated graph families the possible geometries increase considerably. On the other hand, we also add edges and therefore increase the possibilities for minor construction. We believe that the increase in minors overwhelms the increase in geometries. In particular, that trees are indeed the hardest instances, or not far from it.

Conjecture 1. *There is a universal constant $\alpha \geq 1$ such that for every weighted graph $G = (V, E, w)$ and a terminal set $K \subset V$, there is a weighted minor of G supported on K only such that for every $x, y \in K$,*

$$d_G(x,y) \leq d_M(x,y) \leq \alpha \cdot d_G(x,y) \ .$$

Both our framework (`Relaxed-Voronoi`) and the previously used one (`Ball-growing`) proceed by creating random terminal partitions. These partitions are determined using random parameters, which are chosen with no consideration whatsoever of the input graph G. In contrast, the optimal tree algorithm of [Gup01] is a deterministic recursive algorithm which makes decisions after considering the tree structure at hand. It seems that the input-oblivious approach of the `Relaxed-Voronoi` and the `Ball-growing` algorithms will fail to push beyond the $\log k$ upper bound. As a conclusion, input-sensitive approaches seem to be more promising for future attempts to resolve the SPR problem.

In the `Relaxed-Voronoi` algorithm there are two degrees of freedom: choosing the order of terminals, and the magnitude of each terminal. In [Fil19] we choose the order arbitrarily, and the magnitudes randomly with exponential-like distribution. In a follow-up paper with Krauthgamer and Trabelsi [FKT19], we used the `Relaxed-Voronoi` algorithm in order to re-prove Gupta's [Gup01] optimal upper bound of 8. This was done by deterministically choosing order and magnitudes, where the order depends on the graph's geometry. This example demonstrates that one can use the `Relaxed-Voronoi` algorithm also in an input-sensitive manner in order to achieve optimal results.

In the final paper presented in this thesis [FK17], we studied sparsification of binary CSPs with domain of size 2. Our results have been generalized to any finite domain D [BZ19]. As CSPs are broadly used, we believe that these sparsification results will soon find applications. Moreover, it will be very interesting to generalize these results beyond binary. One special case that has been studied is the sparsification of cut edges in hypergraphs [KK15, SY19]. Further, Soma and Yoshida used this hypergraph sparsifier in order to learn and provide succinct representation of sub-modular functions.

We finish with a list of open questions:

- PRIORITIZED JL: Recently, in [MMMR18, NN18] a terminal version of the JL lemma was constructed. Specifically, given a set $K \subseteq \mathbb{R}^d$ of k terminals, an embedding $\phi : \mathbb{R}^d \to \mathbb{R}^{O(\log k/\epsilon^2)}$ with terminal distortion $1+\epsilon$ was constructed. We would like to get a similar result with prioritized dimension. Specifically, given a set $X \subseteq \mathbb{R}^d$ with priority ordering $x_1, x_2, \dots, x_n$, we would like to create an embedding $\psi : X \to \ell_2$ with distortion $1+\epsilon$ such that $\psi(x_j)$ can be non zero only in the first $\alpha(j)$ coordinates. For which functions $\alpha : \mathbb{N} \to \mathbb{N}$ is this possible? Clearly for $\alpha(j) = j-1$ we can even get isometry (using rotations). Ideally, we would like to get $\alpha(j) = O(\log j/\epsilon^2)$.

- PRIORITZED SPANNER: In [BFN19] we constructed a spanner with prioritized distortion $\tilde{O}(\log j)$ and constant lightness. Could we reduce the prioritized distortion to a clean $O(\log j)$?

- STOCHASTIC EMBEDDING INTO SPANNING TREES: It is known how to embed every n-point metric space via stochastic embedding into tree metrics with expected distortion $O(\log n)$. When the input is a weighted graph, it might be beneficial to embed it into distribution of spanning trees, instead of just arbitrary ones. However, here it is only known how to embed with expected distortion $\tilde{O}(\log n)$ [AN12]. Could we embed into a distribution of trees with expected distortion $O(\log n)$?

- SPR: Prove/disprove Conjecture 1. As making progress on the conjecture might be hard, we present several simpler problems.

  - EXPECTED DISTORTION: What distortion parameters could we achieve by stochastic embedding into a distribution of minors, instead of a single embedding? Currently, for general graphs the state of the art for usual (worst-case) distortion, and expected distortion for the SPR problem are the same, $O(\log k)$ upper bound and $\Omega(1)$ lower bound. What are the right bounds for expected distortion for the SPR problem? For planar graphs for example an $O(1)$ distortion is known. Could we achieve similar bound for general graphs?

  - Special graph families: [BG08] showed a constant distortion for the SPR problem on outer-planar graphs. It will be very interesting to achieve better upper bounds for planar graphs, and more generally for minor-free graphs, bounded treewidth graphs etc. In the expected distortion regime, an $O(1)$ upper bound is already known [EGK+14] for these families. Possibly one can use the `Relaxed-Voronoi` algorithm with a clever choice of order and magnitudes in order to achieve such results.

- BEYOND BINARY CSP'S: In our paper on CSP sparsification [FK17], we characterized which binary predicates with domain of size 2 are sparsifiable. In a recent follow-up [BZ19], this result was generalized to arbitrary finite domains. However, the case of arity 3 and beyond is open. Could we generalize the results to higher arities?

References

[ABC+05] Ittai Abraham, Yair Bartal, Hubert T.-H. Chan, Kedar Dhamdhere, Anupam Gupta, Jon M. Kleinberg, Ofer Neiman, and Aleksandrs Slivkins. Metric embeddings with relaxed guarantees. In *FOCS*, pages 83–100. IEEE Computer Society, 2005. 5

[ABN11] Ittai Abraham, Yair Bartal, and Ofer Neiman. Advances in metric embedding theory. *Advances in Mathematics*, 228(6):3026 – 3126, 2011. 5, 7, 8

[ABN15] Ittai Abraham, Yair Bartal, and Ofer Neiman. Embedding metrics into ultrametrics and graphs into spanning trees with constant average distortion. *SIAM J. Comput.*, 44(1):160–192, 2015. 8

[ACE+18] Ittai Abraham, Shiri Chechik, Michael Elkin, Arnold Filtser, and Ofer Neiman. Ramsey spanning trees and their applications. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2018, New Orleans, LA, USA, January 7-10, 2018*, pages 1650–1664, 2018. 11

[ACK+16] Alexandr Andoni, Jiecao Chen, Robert Krauthgamer, Bo Qin, David P. Woodruff, and Qin Zhang. On sketching quadratic forms. In *Innovations in Theoretical Computer Science*, ITCS'16, pages 311–319. ACM, 2016. 4

[ADD+93] I. Althöfer, G. Das, D. P. Dobkin, D. Joseph, and J. Soares. On sparse spanners of weighted graphs. *Discrete & Computational Geometry*, 9:81–100, 1993. 3

[ADF+19] Stephen Alstrup, Søren Dahlgaard, Arnold Filtser, Morten Stöckel, and Christian Wulff-Nilsen. Constructing light spanners deterministically in near-linear time. In *27th Annual European Symposium on Algorithms, ESA 2019, September 9-11, 2019, Munich/Garching, Germany.*, pages 4:1–4:15, 2019. 10, 11

[AFGN18] Ittai Abraham, Arnold Filtser, Anupam Gupta, and Ofer Neiman. Metric embedding via shortest path decompositions. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2018, Los Angeles, CA, USA, June 25-29, 2018*, pages 952–963, 2018. 11

[AG09] Kook Jin Ahn and Sudipto Guha. Graph sparsification in the semi-streaming model. In *36th International Colloquium on Automata, Languages and Programming*, ICALP '09, pages 328–338. Springer-Verlag, 2009. 4

[AGG+14] Ittai Abraham, Cyril Gavoille, Anupam Gupta, Ofer Neiman, and Kunal Talwar. Cops, robbers, and threatening skeletons: padded decomposition for minor-free graphs. In *Symposium on Theory of Computing, STOC 2014, New York, NY, USA, May 31 - June 03, 2014*, pages 79–88, 2014. 6, 12

[AN12] Ittai Abraham and Ofer Neiman. Using petal-decompositions to build a low stretch spanning tree. In *Proceedings of the 44th Symposium on Theory of Computing Conference, STOC 2012, New York, NY, USA, May 19 - 22, 2012*, pages 395–406, 2012. 2, 14

[Bar96a] Yair Bartal. Probabilistic approximation of metric spaces and its algorithmic applications. In *37th Annual Symposium on Foundations of Computer Science (Burlington, VT, 1996)*, pages 184–193. IEEE Comput. Soc. Press, Los Alamitos, CA, 1996. 7

[Bar96b] Yair Bartal. Probabilistic approximations of metric spaces and its algorithmic applications. In *FOCS*, pages 184–193, 1996. 2

[BBMN11] Nikhil Bansal, Niv Buchbinder, Aleksander Madry, and Joseph Naor. A polylogarithmic-competitive algorithm for the k-server problem. In *Foundations of Computer Science (FOCS), 2011 IEEE 52nd Annual Symposium on*, pages 267 –276, oct. 2011. 1

[BFN16] Yair Bartal, Arnold Filtser, and Ofer Neiman. On notions of distortion and an almost minimum spanning tree with constant average distortion. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10-12, 2016*, pages 873–882, 2016. The embedding of β-decomposable metrics appears in the full version http://arxiv.org/abs/1609.08801. 7

[BFN19] Yair Bartal, Arnold Filtser, and Ofer Neiman. On notions of distortion and an almost minimum spanning tree with constant average distortion. *J. Comput. Syst. Sci.*, 105:116–129, 2019. 6, 7, 14

[BG08] A. Basu and A. Gupta. Steiner point removal in graph metrics. Unpublished Manuscript, available from http://www.math.ucdavis.edu/~abasu/papers/SPR.pdf, 2008. 4, 14

[BK96] A. A. Benczúr and D. R. Karger. Approximating s-t minimum cuts in $\tilde{O}(n^2)$ time. In *28th Annual ACM Symposium on Theory of Computing*, pages 47–55. ACM, 1996. 4

[BK02] András A. Benczúr and David R. Karger. Randomized approximation schemes for cuts and flows in capacitated graphs. *CoRR*, cs.DS/0207078, 2002. 4

[Bou85] J. Bourgain. On lipschitz embedding of finite metric spaces in hilbert space. *Israel Journal of Mathematics*, 52(1-2):46–52, 1985. 1

[BS07] Surender Baswana and Sandeep Sen. A simple and linear time randomized algorithm for computing sparse spanners in weighted graphs. *Random Structures & Algorithms*, 30(4):532–563, 2007. See also ICALP'03. 11

[BSS14] Joshua D. Batson, Daniel A. Spielman, and Nikhil Srivastava. Twice-Ramanujan sparsifiers. *SIAM Review*, 56(2):315–334, 2014. 4

[BZ19] Silvia Butti and Stanislav Zivny. Sparsification of binary csps. In *36th International Symposium on Theoretical Aspects of Computer Science, STACS 2019, March 13-16, 2019, Berlin, Germany*, pages 17:1–17:8, 2019. 9, 13, 14

[CDG06] Hubert T.-H. Chan, Michael Dinitz, and Anupam Gupta. Spanners with slack. In *Algorithms - ESA 2006, 14th Annual European Symposium, Zurich, Switzerland, September 11-13, 2006, Proceedings*, pages 196–207, 2006. 5

[CDNS92] B. Chandra, G. Das, G. Narasimhan, and J. Soares. New sparseness results on graph spanners. In *Proc. of 8th SOCG*, pages 192–201, 1992. 3

[CGH16] Yun Kuen Cheung, Gramoz Goranci, and Monika Henzinger. Graph minors for preserving terminal distances approximately - lower and upper bounds. In *43rd International Colloquium on Automata, Languages, and Programming, ICALP 2016, July 11-15, 2016, Rome, Italy*, pages 131:1–131:14, 2016. 4

[CGN+03] Chandra Chekuri, Anupam Gupta, Ilan Newman, Yuri Rabinovich, and Alistair Sinclair. Embedding k-outerplanar graphs into ℓ_1. In *SODA '03: Proceedings of the fourteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 527–536. Society for Industrial and Applied Mathematics, 2003. 5

[Che13] Shiri Chechik. Compact routing schemes with improved stretch. In *ACM Symposium on Principles of Distributed Computing, PODC '13, Montreal, QC, Canada, July 22-24, 2013*, pages 33–41, 2013. 3

[Che14] Shiri Chechik. Approximate distance oracles with constant query time. In *Proceedings of the 46th Annual ACM Symposium on Theory of Computing*, STOC '14, pages 654–663, New York, NY, USA, 2014. ACM. 2

[Che15] Shiri Chechik. Approximate distance oracles with improved bounds. In *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing, STOC 2015, Portland, OR, USA, June 14-17, 2015*, pages 1–10, 2015. 2

[Che18] Yun Kuen Cheung. Steiner point removal - distant terminals don't (really) bother. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2018, New Orleans, LA, USA, January 7-10, 2018*, pages 1353–1360, 2018. 4, 8

[CJLV08] Amit Chakrabarti, Alexander Jaffe, James R. Lee, and Justin Vincent. Embeddings of topological graphs: Lossy invariants, linearization, and 2-sums. In *49th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2008, October 25-28, 2008, Philadelphia, PA, USA*, pages 761–770, 2008. 5

[CKR01] Gruia Calinescu, Howard Karloff, and Yuval Rabani. Approximation algorithms for the 0-extension problem. In *Proceedings of the twelfth annual ACM-SIAM symposium on Discrete algorithms*, SODA '01, pages 8–16, Philadelphia, PA, USA, 2001. Society for Industrial and Applied Mathematics. 12

[CW18] Shiri Chechik and Christian Wulff-Nilsen. Near-optimal light spanners. *ACM Trans. Algorithms*, 14(3):33:1–33:15, 2018. 3, 10, 11

[CXKR06] T.-H. Chan, Donglin Xia, Goran Konjevod, and Andrea Richa. A tight lower bound for the steiner point removal problem on trees. In *Proceedings of the 9th International Conference on Approximation Algorithms for Combinatorial Optimization Problems, and 10th International Conference on Randomization and Computation*, APPROX'06/RANDOM'06, pages 70–81, Berlin, Heidelberg, 2006. Springer-Verlag. 4, 13

[dCHS11] Marcel K. de Carli Silva, Nicholas J. A. Harvey, and Cristiane M. Sato. Sparse sums of positive semidefinite matrices. *CoRR*, abs/1107.0088, 2011. 4

[DHN93] Gautam Das, Paul J. Heffernan, and Giri Narasimhan. Optimally sparse spanners in 3-dimensional euclidean space. In *Proceedings of the Ninth Annual Symposium on Computational GeometrySan Diego, CA, USA, May 19-21, 1993*, pages 53–62, 1993. 3

[EEST05] Michael Elkin, Yuval Emek, Daniel A. Spielman, and Shang-Hua Teng. Lower-stretch spanning trees. In *STOC '05: Proceedings of the thirty-seventh annual ACM symposium on Theory of computing*, pages 494–503, New York, NY, USA, 2005. ACM Press. 2

[EFN15] Michael Elkin, Arnold Filtser, and Ofer Neiman. Prioritized metric structures and embedding. In *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing, STOC 2015, Portland, OR, USA, June 14-17, 2015*, pages 489–498, 2015. 6

[EFN17] Michael Elkin, Arnold Filtser, and Ofer Neiman. Terminal embeddings. *Theor. Comput. Sci.*, 697:1–36, 2017. 5

[EFN18] Michael Elkin, Arnold Filtser, and Ofer Neiman. Prioritized metric structures and embedding. *SIAM J. Comput.*, 47(3):829–858, 2018. 6, 7, 8

[EGK+14] Matthias Englert, Anupam Gupta, Robert Krauthgamer, Harald Räcke, Inbal Talgam-Cohen, and Kunal Talwar. Vertex sparsifiers: New results from old techniques. *SIAM J. Comput.*, 43(4):1239–1262, 2014. 4, 12, 14

[EN17] Michael Elkin and Ofer Neiman. Efficient algorithms for constructing very sparse spanners and emulators. *CoRR*, abs/1607.08337, Version 2, 2017. 11

[EN18] Michael Elkin and Ofer Neiman. Near isometric terminal embeddings for doubling metrics. In *34th International Symposium on Computational Geometry, SoCG 2018, June 11-14, 2018, Budapest, Hungary*, pages 36:1–36:15, 2018. 5

[ENS14] Michael Elkin, Ofer Neiman, and Shay Solomon. Light spanners. In *Proc. of 41th ICALP*, pages 442–452, 2014. 3

[Erd64] Paul Erdős. Extremal problems in graph theory. In *Proc. of Sympos. Smolenice*, pages 29–36, 1964. 3

[ES16] Michael Elkin and Shay Solomon. Fast constructions of lightweight spanners for general graphs. 12(3):29:1–29:21, 2016. See also SODA'13. 11

[FHHP11] Wai Shing Fung, Ramesh Hariharan, Nicholas J.A. Harvey, and Debmalya Panigrahi. A general framework for graph sparsification. In *43rd Annual ACM Symposium on Theory of Computing*, pages 71–80. ACM, 2011. 4

[Fil18] Arnold Filtser. Steiner point removal with distortion $O(\log k)$. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2018, New Orleans, LA, USA, January 7-10, 2018*, pages 1361–1373, 2018. 8

[Fil19] Arnold Filtser. Steiner point removal with distortion o(log k) using the relaxed-voronoi algorithm. *SIAM J. Comput.*, 48(2):249–278, 2019. 8, 12, 13

[FK17] Arnold Filtser and Robert Krauthgamer. Sparsification of two-variable valued constraint satisfaction problems. *SIAM J. Discrete Math.*, 31(2):1263–1276, 2017. 9, 13, 14

[FKT19] Arnold Filtser, Robert Krauthgamer, and Ohad Trabelsi. Relaxed voronoi: A simple framework for terminal-clustering problems. In *2nd Symposium on Simplicity in Algorithms, SOSA@SODA 2019, January 8-9, 2019 - San Diego, CA, USA*, pages 10:1–10:14, 2019. 12, 13

[FN18] Arnold Filtser and Ofer Neiman. Light spanners for high dimensional norms via stochastic decompositions. In *26th Annual European Symposium on Algorithms, ESA 2018, August 20-22, 2018, Helsinki, Finland*, pages 29:1–29:15, 2018. 10

[FRT04] Jittat Fakcharoenphol, Satish Rao, and Kunal Talwar. A tight bound on approximating arbitrary metrics by tree metrics. *J. Comput. Syst. Sci.*, 69(3):485–497, November 2004. 2

[FS16] Arnold Filtser and Shay Solomon. The greedy spanner is existentially optimal. In *Proceedings of the 2016 ACM Symposium on Principles of Distributed Computing, PODC 2016, Chicago, IL, USA, July 25-28, 2016*, pages 9–17, 2016. 3, 9, 10, 11

[GLN02] Joachim Gudmundsson, Christos Levcopoulos, and Giri Narasimhan. Fast greedy algorithms for constructing sparse geometric spanners. *SIAM J. Comput.*, 31(5):1479–1500, 2002. 10

[GNRS04] Anupam Gupta, Ilan Newman, Yuri Rabinovich, and Alistair Sinclair. Cuts, trees and ℓ_1s-embeddings of graphs. *Combinatorica*, 24(2):233–269, 2004. 1, 5, 6

[Got15] Lee-Ad Gottlieb. A light metric spanner. In *Proc. of 56th FOCS*, pages 759–772, 2015. 3, 10

[GPPR01] Cyril Gavoille, David Peleg, Stephane Perennes, and Ran Raz. Distance labeling in graphs. In *Proceedings of the Twelfth Annual Symposium on Discrete Algorithms, January 7-9, 2001, Washington, DC, USA.*, pages 210–219, 2001. 2

[Gup01] Anupam Gupta. Steiner points in tree metrics don't (really) help. In *Proceedings of the Twelfth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '01, pages 220–227, Philadelphia, PA, USA, 2001. Society for Industrial and Applied Mathematics. 4, 12, 13

[HPIS13] Sariel Har-Peled, Piotr Indyk, and Anastasios Sidiropoulos. Euclidean spanners in high dimensions. In *Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '13, pages 804–809. SIAM, 2013. 10

[HST86] Cor A. J. Hurkens, Alexander Schrijver, and Éva Tardos. On fractional multicommodity flows and distance functions. *Discrete Mathematics*, 73:99–109, 1986. 5

[JL84] William Johnson and Joram Lindenstrauss. Extensions of Lipschitz mappings into a Hilbert space. *Contemporary Mathematics*, 26:189–206, 1984. 1

[KK15] Dmitry Kogan and Robert Krauthgamer. Sketching cuts in graphs and hypergraphs. In *Proceedings of the 2015 Conference on Innovations in Theoretical Computer Science, ITCS 2015, Rehovot, Israel, January 11-13, 2015*, pages 367–376, 2015. 4, 9, 14

[KK16] Lior Kamma and Robert Krauthgamer. Metric decompositions of path-separable graphs. *Algorithmica*, pages 1–9, 2016. 12

[KKM+12] Maleq Khan, Fabian Kuhn, Dahlia Malkhi, Gopal Pandurangan, and Kunal Talwar. Efficient distributed approximation algorithms via probabilistic tree embeddings. *Distributed Computing*, 25(3):189–205, 2012. 1

[KKN15] Lior Kamma, Robert Krauthgamer, and Huy L. Nguyen. Cutting corners cheaply, or how to remove steiner points. *SIAM J. Comput.*, 44(4):975–995, 2015. 4, 8

[KL02] David R. Karger and Matthew S. Levine. Random sampling in residual graphs. In *Proceedings of the Symposium on Theory of Computing (STOC)*, pages 63–66, 2002. 4

[Kle02] Philip N. Klein. Preprocessing an undirected planar network to enable fast approximate distance queries. In *Proceedings of the Thirteenth Annual ACM-SIAM Symposium on Discrete Algorithms, January 6-8, 2002, San Francisco, CA, USA.*, pages 820–827, 2002. 2

[KLMN04] Robert Krauthgamer, James R. Lee, Manor Mendel, and Assaf Naor. Measured descent: A new embedding method for finite metrics. In *45th Annual IEEE Symposium on Foundations of Computer Science*, pages 434–443. IEEE, October 2004. 5, 6, 12

[KNZ14] Robert Krauthgamer, Huy L. Nguyen, and Tamar Zondiner. Preserving terminal distances using minors. *SIAM J. Discrete Math.*, 28(1):127–141, 2014. 4

[KP12] Michael Kapralov and Rina Panigrahy. Spectral sparsification via random spanners. In *3rd Innovations in Theoretical Computer Science Conference*, pages 393–398. ACM, 2012. 4

[KSW04] Jon M. Kleinberg, Aleksandrs Slivkins, and Tom Wexler. Triangulation and embedding using small sets of beacons. In *FOCS*, pages 444–453, 2004. 5, 8

[LLR95] Nathan Linial, Eran London, and Yuri Rabinovich. The geometry of graphs and some of its algorithmic applications. *Combinatorica*, 15(2):215–245, 1995. 1, 6

[LN03] James R. Lee and Assaf Naor. Metric decomposition, smooth measures, and clustering. Unpublished Manuscript, available from `https://www.math.nyu.edu/~naor/homepage%20files/cluster.pdf`, 2003. 12

[LR10] James R. Lee and Prasad Raghavendra. Coarse differentiation and multi-flows in planar graphs. *Discrete & Computational Geometry*, 43(2):346–362, 2010. 5

[LS10] James R. Lee and Anastasios Sidiropoulos. Genus and the geometry of the cut graph. In *Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2010, Austin, Texas, USA, January 17-19, 2010*, pages 193–201, 2010. 6

[LS13] James R. Lee and Anastasios Sidiropoulos. Pathwidth, trees, and random embeddings. *Combinatorica*, 33(3):349–374, 2013. 5, 6, 12

[Mad10] Aleksander Madry. Fast approximation algorithms for cut-based problems in undirected graphs. In *Proceedings of the Symposium on Foundations of Computer Science (FOCS)*, pages 245–254. IEEE, 2010. 4

[Mat02] Jiri Matoušek. *Lectures on discrete geometry.* Springer-Verlag, New York, 2002. 1

[MMMR18] Sepideh Mahabadi, Konstantin Makarychev, Yury Makarychev, and Ilya P. Razenshteyn. Nonlinear dimension reduction via outer bi-lipschitz extensions. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2018, Los Angeles, CA, USA, June 25-29, 2018*, pages 1088–1101, 2018. 5, 14

[MN07] Manor Mendel and Assaf Naor. Ramsey partitions and proximity data structures. *Journal of the European Mathematical Society*, 9(2):253–275, 2007. 11

[MPVX15] Gary L. Miller, Richard Peng, Adrian Vladu, and Shen Chen Xu. Improved parallel algorithms for spanners and hopsets. In *Proc. 27th*, pages 192–201, 2015. 11

[NN18] Shyam Narayanan and Jelani Nelson. Optimal terminal dimensionality reduction in euclidean space. *CoRR*, abs/1810.09250, 2018. 5, 14

[NR13] I. Newman and Y. Rabinovich. On multiplicative λ-approximations and some geometric applications. *SIAM Journal on Computing*, 42(3):855–883, 2013. 4

[OS81] Haruko Okamura and P.D. Seymour. Multicommodity flows in planar graphs. *Journal of Combinatorial Theory, Series B*, 31(1):75 – 81, 1981. 5

[Pel99] David Peleg. Proximity-preserving labeling schemes and their applications. In *Graph-Theoretic Concepts in Computer Science, 25th International Workshop, WG '99, Ascona, Switzerland, June 17-19, 1999, Proceedings*, pages 30–41, 1999. 2

[Rab08] Yuri Rabinovich. On average distortion of embedding metrics into the line. *Discrete & Computational Geometry*, 39(4):720–733, 2008. 6

[Rao99] Satish Rao. Small distortion and volume preserving embeddings for planar and Euclidean metrics. In *Proceedings of the Fifteenth Annual Symposium on Computational Geometry, Miami Beach, Florida, USA, June 13-16, 1999*, pages 300–306, 1999. 5, 12

[RTZ05] Liam Roditty, Mikkel Thorup, and Uri Zwick. Deterministic constructions of approximate distance oracles and spanners. In *Proceedings of the 32Nd International Conference on Automata, Languages and Programming*, ICALP'05, pages 261–272, Berlin, Heidelberg, 2005. Springer-Verlag. 11

[She09] Jonah Sherman. Breaking the multicommodity flow barrier for $O(\sqrt{\log n})$-approximations to sparsest cut. In *Proceedings of the Symposium on Foundations of Computer Science (FOCS)*, pages 363–372, 2009. 4

[Smi09] Michiel H. M. Smid. The weak gap property in metric spaces of bounded doubling dimension. In *Efficient Algorithms, Essays Dedicated to Kurt Mehlhorn on the Occasion of His 60th Birthday*, pages 275–289, 2009. 3

[SS11] Daniel A. Spielman and Nikhil Srivastava. Graph sparsification by effective resistances. *SIAM J. Comput.*, 40(6):1913–1926, dec 2011. 4

[ST04a] Yuval Shavitt and Tomer Tankel. Big-bang simulation for embedding network distances in Euclidean space. *IEEE/ACM Trans. Netw.*, 12(6):993–1006, 2004. 1

[ST04b] D. A. Spielman and S.-H. Teng. Nearly-linear time algorithms for graph partitioning, graph sparsification, and solving linear systems. In *36th Annual ACM Symposium on Theory of Computing*, pages 81–90. ACM, 2004. 4

[ST11] Daniel A. Spielman and Shang-Hua Teng. Spectral sparsification of graphs. *SIAM J. Comput.*, 40(4):981–1025, jul 2011. 4

[SY19] Tasuku Soma and Yuichi Yoshida. Spectral sparsification of hypergraphs. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019, San Diego, California, USA, January 6-9, 2019*, pages 2570–2581, 2019. 14

[Tho01] Mikkel Thorup. Compact oracles for reachability and approximate distances in planar digraphs. In *42nd Annual Symposium on Foundations of Computer Science, FOCS 2001, 14-17 October 2001, Las Vegas, Nevada, USA*, pages 242–251, 2001. 2

[TZ01a] M. Thorup and U. Zwick. Approximate distance oracles. In 33^{rd} *Annual ACM Symposium on Theory of Computing (STOC)*, pages 183–192, Hersonissos, Crete, Greece, July 2001. 2

[TZ01b] Mikkel Thorup and Uri Zwick. Compact routing schemes. In *SPAA*, pages 1–10, 2001. 3

[WN13] Christian Wulff-Nilsen. Approximate distance oracles with improved query time. In *Proceedings of the Twenty-Forth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '13. SIAM, 2013. 2

Part II
Prioritized Metric Structures and Embedding

SIAM J. COMPUT.
Vol. 47, No. 3, pp. 829–858

PRIORITIZED METRIC STRUCTURES AND EMBEDDING*

MICHAEL ELKIN†, ARNOLD FILTSER†, AND OFER NEIMAN†

Abstract. Metric data structures (distance oracles, distance labeling schemes, routing schemes) and low-distortion embeddings provide a powerful algorithmic methodology, which has been successfully applied for approximation algorithms [N. Linial, E. London, and Y. Rabinovich, *Combinatorica*, 15 (1995), pp. 215–245], online algorithms [N. Bansal et al., *Proceedings of the* 52*th Annual IEEE Symposium on Foundations of Computer Science, FOCS* '08, IEEE Computer Society, Washington, DC, 2011, pp. 267–276], distributed algorithms [M. Khan et al., *Distrib. Comput.*, 25 (2012), pp. 189–205], and for computing sparsifiers [Y. Shavitt and T. Tankel, *IEEE/ACM Trans. Netw.*, 12 (2004), pp. 993–1006]. However, this methodology appears to have a limitation: the worst-case performance inherently depends on the cardinality of the metric, and one could not specify in advance which vertices/points should enjoy a better service (i.e., stretch/distortion, label size/dimension) than that given by the worst-case guarantee. In this paper we alleviate this limitation by devising a suite of *prioritized* metric data structures and embeddings. We show that given a priority ranking $(x_1, x_2, \ldots, x_n)$ of the graph vertices (resp., metric points) one can devise a metric data structure (resp., embedding) in which the stretch (resp., distortion) incurred by any pair containing a vertex x_j will depend on the rank j of the vertex. We also show that other important parameters, such as the label size and (in some sense) the dimension, may depend only on j. In some of our metric data structures (resp., embeddings) we achieve both prioritized stretch (resp., distortion) and label size (resp., dimension) *simultaneously*. The worst-case performance of our metric data structures and embeddings is typically asymptotically no worse than of their nonprioritized counterparts.

Key words. metric embedding, distance oracles, routing, priorities

AMS subject classifications. 68W01, 68P05

DOI. 10.1137/17M1118749

1. Introduction. The celebrated distance oracle of Thorup and Zwick [TZ05] enables one to preprocess an undirected weighted n-vertex graph $G = (V, E)$ so as to produce a data structure (also known as *distance oracle*) of size $O(t \cdot n^{1+1/t})$ (for a parameter $t = 1, 2, \ldots$) that supports distance queries between pairs $u, v \in V$ in time $O(t)$ per query. (The query time was recently improved to $O(1)$ by [Che14, Wul13], and the size to $O(n^{1+1/t})$ by [Che15].) The distance estimates provided by the oracle are within a factor of $2t-1$ from the actual distance $d_G(u, v)$ between u and v in G. The approximation factor ($2t-1$ in this case) is called the *stretch.* Distance oracles can serve as an example of a *metric data structure*; other very well-studied examples include *distance labeling* [Pel99, GPPR01] and *routing* [TZ01, AP92]. Thorup–Zwick's oracle can also be converted into a distance-labeling scheme: each vertex is assigned a label of size $O(n^{1/t} \cdot \log^{1-1/t} n)$ so that given labels of u and v the query algorithm can provide a $(2t-1)$-approximation of $d_G(u, v)$. Moreover, the oracle also gives rise to a routing scheme [TZ01] that exhibits a similar trade-off.

A different but closely related thread of research concerns *low-distortion embeddings.* A celebrated theorem of Bourgain [Bou86] asserts that any n-point metric (X, d) can be embedded into an $O(\log n)$-dimensional Euclidean space with *distortion*

*Received by the editors February 27, 2017; accepted for publication (in revised form) February 9, 2018; published electronically June 14, 2018. A preliminary version of this paper was published in STOC'15, ACM, New York, 2015, pp. 489–498 [EFN15].

http://www.siam.org/journals/sicomp/47-3/M111874.html

Funding: The first author's research was supported by the ISF grant (724/15). The third author's research was supported in part by ISF grant (523/12) and by BSF grant 2015813.

†Department of Computer Science, Ben-Gurion University of the Negev, Beer-Sheva, Israel (elkinm@cs.bgu.ac.il, arnoldf@cs.bgu.ac.il, neimano@cs.bgu.ac.il).

$O(\log n)$. (Roughly speaking, distortion and stretch are the same thing. See section 2 for formal definitions.) Fakcharoenphol, Rao, and Talwar [FRT04] (following Bartal [Bar96, Bar98]) showed that any metric (X, d) embeds into a distribution over trees (in fact, ultrametrics) with expected distortion $O(\log n)$.

These (and many other) important results are not only appealing from a mathematical perspective, but they also were found extremely useful for numerous applications in theoretical computer science and beyond [LLR95, BBMN11, KKM+12, ST04]. A natural disadvantage is the dependence of all the relevant parameters on n, the cardinality of the input graph/metric. However, all these results are either completely tight, or very close to being completely tight. In order to address this issue, metric data structures and embeddings in which some pairs of vertices/points enjoy better stretch/distortion or with improved label size/dimension were developed. Specifically, [KSW09, ABC+05, ABN11, CDG06] studied embeddings and distance oracles in which the distortion/stretch of at least $1 - \epsilon$ fraction of the pairs is improved as a function of ϵ, either for a fixed ϵ or for all $\epsilon \in [0, 1]$ simultaneously (e.g., for a fixed ϵ, embeddings into Euclidean space of dimension $O(\log 1/\epsilon)$ with distortion $O(\log(1/\epsilon))$, or a distance oracle with stretch $2\lceil t \cdot \frac{\log(2/\epsilon)}{\log n} \rceil + 1$ for $1 - \epsilon$ fraction of the pairs). Also, [ABN07, SS09, AC14] devised embeddings and distance oracles that provide distortion/stretch $O(\log k)$ for all pairs (x, y) of points such that y is among the k closest points to x, and distance labeling schemes that support queries only between k-nearest neighbors, in which the label size depends only on k rather than n.

An inherent shortcoming of these results is, however, that the pairs that enjoy better than worst-case distortion cannot be specified in advance. In this paper we alleviate this shortcoming and devise a suite of prioritized metric data structures and low-distortion embeddings. Specifically, we show that one can order the graph vertices $V = (x_1, \ldots, x_n)$ *arbitrarily in advance*, and devise metric data structures (i.e., oracles/labelings/routing schemes) that, for a parameter $t = 1, 2, \ldots$, provide stretch $2\lceil t \cdot \frac{\log j}{\log n} \rceil - 1$ (instead of $2t - 1$) for *all pairs involving* x_j,[1] while using the same space as corresponding nonprioritized data structures! In some cases the label size can be simultaneously improved for the high priority points, as described in the following.

The same phenomenon occurs for low-distortion embeddings. We devise an embedding of general metrics into an $O(\log n)$-dimensional Euclidean space that provides *prioritized distortion* $O(\log j \cdot (\log \log j)^{1/2+\epsilon})$, for any constant $\epsilon > 0$ (i.e., the distortion for all pairs containing x_j is $O(\log j \cdot (\log \log j)^{1/2+\epsilon})$). Similarly, our embedding into a distribution of trees provides prioritized expected distortion $O(\log j)$.

We introduce a novel notion of *improved dimension* for high priority points. In general we cannot expect that the dimension of a Euclidean embedding with low distortion (even prioritized) will be small (as Euclidean embedding into dimension D has worst-case distortion of $\Omega(n^{1/D} \cdot \log n)$ for some metrics [ABN11]). What we can offer is an embedding in which the high ranked points have only a few "active" coordinates. That is, only the first $O(\mathrm{poly}(\log j))$ coordinates in the image of x_j will be nonzero, while the distortion is also bounded by $O(\mathrm{poly}(\log j))$. This could be useful in a setting where the high ranked points participate in numerous computations, then since representing these points requires very few coordinates, we can store many of

[1] In the case $j = 1$, the stretch is 1. For ease of presentation, we ignore this special case in the statement of the results—the stretch/distortion for x_1 will always be at most the value guaranteed for x_2. (In the technical sections we do provide a separate analysis for x_1 when needed.)

them in the cache or other high speed memory. We remark that our framework is *the first* which allows simultaneously improved distortion and dimension (or improved stretch and label size) for the high priority points, while providing a meaningful guarantee for all pairs.

We have a construction of prioritized distance oracles that exhibits a qualitatively different behavior than our aforementioned oracles. Specifically, we devise a distance oracle with space $O(n \log\log n)$ (resp., $O(n\log^* n)$) and prioritized stretch $O(\frac{\log n}{\log(n/j)})$ (resp., $2^{O(\frac{\log n}{\log(n/j)})}$). Observe that as long as $j < n^{1-\epsilon}$ for any fixed $\epsilon > 0$, the prioritized stretch of both these oracles is $O(1)$. The query time is $O(1)$. These oracles are, however, not path reporting (a path-reporting oracle can return an actual approximate shortest path in the graph, in time proportional to its length). We also devise a path-reporting prioritized oracle, which was mentioned above: it has space $O(t \cdot n^{1+1/t})$, stretch $2\lceil t \cdot \frac{\log j}{\log n}\rceil - 1$, and query time $O(t \cdot \frac{\log j}{\log n})$.

This second oracle can be distributed as a labeling scheme, in which not only the stretch $2\lceil t \cdot \frac{\log j}{\log n}\rceil - 1$ is prioritized, but also the label size is smaller for high priority points: it is $O(n^{1/t} \cdot \log j)$ rather than the nonprioritized $O(n^{1/t} \cdot \log n)$. Our routing scheme has prioritized stretch $4\lceil t \cdot \frac{\log j}{\log n}\rceil - 1$ (instead of $4t-5$), the routing tables have size $O(n^{1/t} \cdot \log j)$ (instead of $O(n^{1/t} \cdot \log n)$), and labels have size $O(\log j \cdot \lceil t\frac{\log j}{\log n}\rceil)$ (instead of $O(t \cdot \log n)$).

We also consider the dual setting in which the stretch is fixed, and label size $\lambda(j)$ of x_j is smaller when $j \ll n$. The function $\lambda(j)$ will be called *prioritized label size*. Specifically, with prioritized label size $O(j^{1/t} \cdot \log j)$ we can have stretch $2t-1$. For certain points on the trade-off curve we can even have both stretch and label size prioritized simultaneously! In particular, a variant of our distance labeling scheme provides a prioritized stretch $2\lceil \log j\rceil - 1$ and prioritized label size $O(\log j)$. For routing we have similar guarantees independent of n. We also devise a distance labeling scheme for graphs that exclude a fixed minor with stretch $1+\epsilon$ and prioritized label size $O(1/\epsilon \cdot \log j)$ (extending [AG06, Tho01]).

Another notable result in this context is our prioritized embedding into a *single tree*. It is well known that any metric can be embedded into a single dominating tree with linear distortion, and that it is tight [RR98]. We show that any n-point metric (X, d) enjoys an embedding into a single dominating tree with prioritized distortion $\alpha(j)$ *if and only if* the sum of reciprocals $\sum_{j=1}^{\infty} 1/\alpha(j)$ converges. In particular, prioritized distortion $\alpha(j) = j \cdot \log j \cdot (\log\log j)^{1.01}$ is admissible, while $\alpha(j) = j \cdot \log j \cdot \log\log j$ is not, i.e., both our upper and lower bounds are tight. This lower bounds stands out as it shows that it is not always possible to replace nonprioritized distortion of $\alpha(n)$ by a prioritized distortion $\alpha(j)$. For single-tree embedding the nonprioritized distortion is linear, while the prioritized one is provably superlinear.

1.1. Overview of techniques. We elaborate briefly on the methods used to obtain our results.

Distance oracles, distance labeling, and routing. We have two basic techniques for obtaining distance oracles with prioritized stretch. The first one is manifested in Theorem 5, and the idea is as follows: partition the vertices into sets according to their priority, and for each set $K \subseteq V$, apply as a black box a known distance oracle on K, while for the other vertices store the distance to their nearest neighbor in K. We show that the stretch of pairs in $K \times V$ is only a factor of 2 worse than the one guaranteed for $K \times K$. Furthermore, we exploit the fact that for sets K of small size, we can afford a very small stretch and still maintain a small space. The exact choice of the partitions enables a range of trade-offs between space and prioritized stretch.

Our second technique for an oracle with prioritized stretch, used in Theorem 6, is based on a non-black-box variation of the [TZ05] oracle. In their construction for stretch $2t-1$, a (nonincreasing) sequence of $t-1$ sets is generated by repeated random sampling. We show that if a vertex is chosen i times, then the query algorithm can be changed to improve the stretch from $2t-1$ to $2(t-i)-1$, for *any pair* containing such a vertex. This observation only shows that there exists a priority ranking for which the oracle has the required prioritized stretch. In order to handle *any* given ranking, we alter the construction by forcing high ranked elements to be chosen numerous times, and show that this increases the space usage by at most a factor of 2.

In order to build a distance labeling scheme out of their distance oracle, [TZ05] pay an overhead of $O(\log^{1-1/t} n)$ in the label size (which essentially comes from applying concentration bounds). Attempting to circumvent this logarithmic dependence on n, in Theorem 7 we give a different bound on the deviation probability that depends on the priority ranking of the point. Thus the overhead in the label size for the jth point in the ranking is only $O(\log j)$. To derive our result in Theorem 8, which has fixed stretch $2t-1$ for all pairs, but fully prioritized label size $O(j^{1/t} \log j)$, we combine this probabilistic argument with an iterative application of a *source restricted* distance labeling of [RTZ05].

Most results on distance labeling for bounded treewidth graphs, planar graphs, and graphs excluding a fixed minor, are based on recursively partitioning the graph into small pieces using small separators (as in [LT79]). The label of a vertex essentially consists of the distances to (some of) the vertices in the separator. In order to obtain prioritized label size, such as those given in Theorems 10 and 11, high ranked vertices should participate in few iterations. To this end, we define multiple phases of applying separators, where each phase tries to separate only a certain subset of the vertices (starting with the highest ranked, and finishing in the lowest). This way high ranked vertices will belong to a separator after a few levels, thus their label will be short.

Tree routing of [TZ01] is based on categorizing tree vertices as either heavy or light, depending on the size of their subtree. Our prioritized tree routing assigns each vertex a weight which depends on its priority, and a vertex is heavy if the sum of weights of its descendants is sufficiently large. This idea paves the way to our prioritized routing scheme for general graphs as well.

Embeddings. It is folklore that a metric minimum spanning tree (henceforth, MST) achieves distortion $n-1$. For our prioritized embedding of general metrics (X, d) into a single tree we consider a complete graph $G = (X, \binom{X}{2})$ with weight function that depends on the priority ranking. Specifically, edges incident on high priority points get higher weights. We then compute an MST in this (generally nonmetric) graph, and show that, given a certain convergence condition on the priority ranking, this MST provides a desired prioritized single-tree embedding. Remarkably, we also show that when this condition is not met, no such an embedding is possible even for the metric induced by C_n. Hence this embedding is tight.

Our probabilistic embedding into trees with prioritized expected distortion in Theorem 4 is based on the construction of [FRT04]. The method of [FRT04] involves sampling a random permutation and a random radius, then using these to create a hierarchical partitioning of the metric from which a tree is built. We make the observation that, in some sense, the expected distortion of a point depends on its position in the permutation. Rather than choosing a permutation uniformly at random, we choose one which is strongly correlated with the given priority ranking. One must be careful to allow sufficient randomness in the permutation choice so that the analysis can still go through, while guaranteeing that high ranked points will appear in the first positions of the permutation.

The embedding of Theorem 14 for arbitrary metrics (X, d) into Euclidean space (or any ℓ_p space) with prioritized distortion uses similar ideas. We partition the points into sets according to the priorities; for every such a subset K apply as a black box the embedding of [Bou85]. We show that since the embedding has certain properties, it can be extended in a Lipschitz manner to all of the metric, while having distortion guarantee for any pair in $K \times X$.

The result of Theorem 15, which gives prioritized distortion and dimension, is more technically involved. In order to ensure that high priority points are mapped to the zero vector in the embeddings tailored for the lower priority points, we change Bourgain's embedding, which is defined as distances to randomly chosen sets. Roughly speaking, when creating the embedding for a set K, we add all the higher ranked points to the random sets. As a result, the original analysis does not apply directly, and we turn to a subtle case analysis to bound the distortion; see section 8.2 for more details.

Subsequent work. Following our work, [BFN16] exhibited a tight connection between embeddings with prioritized distortion and a certain type of scaling distortion called *coarse scaling distortion*. Using this connection and a result of [ABN11], [BFN16] showed an embedding of general metrics into an $O(\log n)$-dimensional Euclidean space (or any ℓ_p space) with asymptotically optimal prioritized distortion $O(\log j)$, improving our bound of $O(\log j(\log\log j)^{1/2+\epsilon})$, for any $\epsilon > 0$.

1.2. Organization. After a few preliminary definitions, we show the single-tree prioritized embedding in section 3, and the probabilistic version in section 4. In section 5 we discuss our prioritized distance oracles, and in section 6 the prioritized labeling schemes. The prioritized routing is shown in section 7. Finally, in section 8 we present our prioritized embedding results into normed spaces.

2. Preliminaries. Throughout the paper, all logarithms are in base 2. All the graphs $G = (V, E)$ we consider are undirected and weighted. Let $x_1, \ldots, x_n \in V$ be a priority ranking of the vertices. Let d_G be the shortest path metric on G, and let $\alpha, \beta : [n] \to \mathbb{R}_+$ be a monotone nondecreasing functions.

A distance oracle for a graph G is a succinct data structure that can approximately report distances between vertices of G. The parameters of this data structure we will care about are its space, query time, and stretch factor. We always measure the space of the oracle as the number of words needed to store it (where each word is $O(\log n)$ bits). The oracle has *prioritized stretch* $\alpha(j)$ if for any $1 \le j < i \le n$, when queried for x_j, x_i the oracle reports a distance $\tilde{d}(x_j, x_i)$ such that

$$d_G(x_j, x_i) \le \tilde{d}(x_j, x_i) \le \alpha(j) \cdot d_G(x_j, x_i) \ .$$

Some oracles can be distributed as a labeling scheme, where each vertex is given a short label, and the approximate distance between two vertices should be computed by inspecting their labels alone. We say that a labeling scheme has prioritized label size $\beta(j)$ if for every $j \in [n]$, the label of x_j consists of at most $\beta(j)$ words. See section 7 for the precise settings of routing that we consider.

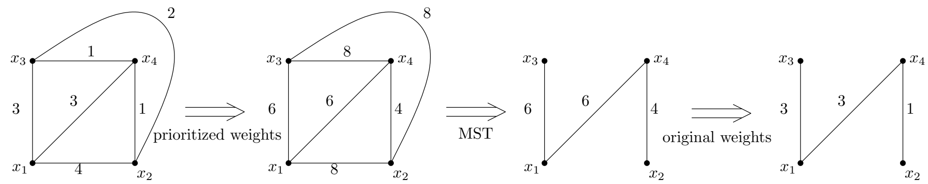

FIG. 1. *An illustration for the algorithm presented during the proof of Theorem* 1. *We are given a metric space over* $X = \{x_1, x_2, x_3, x_4\}$*, with the function* $\alpha(1) = 2, \alpha(2) = 4, \alpha(3) = 8, \alpha(4) = 16$*. In the first step we assign new weights over the edges, then find an MST in the new graph and, finally, restore the original weights. For example the original distance between* x_2, x_3 *was* 2*, while in the returned tree the distance is* 7*. Hence the pair* x_2, x_3 *suffers distortion* $3.5 < 4$.

Let (X, d_X) be a finite metric space, and let $x_1, \ldots, x_n$ be a priority ranking of the points in X. Given a target metric (Y, d_Y), and a noncontractive map $f : X \to Y$,[2] we say that f has *priority distortion* $\alpha(j)$ if for all $1 \le j < i \le n$,

$$d_Y(f(x_j), f(x_i)) \le \alpha(j) \cdot d_X(x_j, x_i) \ .$$

Similarly, if $f : X \to Y$ is nonexpansive, then it has priority distortion $\alpha(j)$ if for all $1 \le j < i \le n$, $d_Y(f(x_j), f(x_i)) \ge d_X(x_j, x_i)/\alpha(j)$. For probabilistic embedding, we require that each map in the support of the distribution is noncontractive, and the prioritized bound on the distortion holds in expectation.

In the special case that the target metric is a normed space, we say that the embedding has *prioritized dimension* $\beta(j)$ if for every $j \in [n]$, only the first $\beta(j)$ coordinates in $f(x_j)$ may be nonzero.

3. Single-tree embedding with prioritized distortion. In this section we show tight bounds on the priority distortion for an embedding into a single tree. The bounds are somewhat nonstandard, as they are not attained for a single specific function, but rather for the following family of functions. Define Φ to be the family of functions $\alpha : \mathbb{N} \to \mathbb{R}_+$ that satisfy the following properties:

- α is nondecreasing.
- $\sum_{i=1}^{\infty} 1/\alpha(i) \le 1$.

3.1. Upper bound.

THEOREM 1. *For any finite metric space* (X, d) *and any* $\alpha \in \Phi$*, there is a (noncontractive) embedding of* X *into a single tree with priority distortion* $2\alpha(j)$.

Proof. Let $x_1, \ldots, x_n$ be the priority ranking of X, and let $G = (X, E)$ be the complete graph on X. For $e = \{u, v\} \in E$, let $\ell(e) = d(u, v)$. We also define the following (prioritized) weights $w : E \to \mathbb{R}$, for any $1 \le j < i \le n$ the edge $e = \{x_j, x_i\}$ will be given the weight $w(e) = \alpha(j) \cdot \ell(e)$. Observe that the w weights on G may not satisfy the triangle inequality. Let T be the MST of (X, E, w) (this tree is formed by iteratively removing the heaviest edge from a cycle). Finally, return the tree T with the edges weighted by ℓ. We claim that this tree has priority distortion $\alpha(j)$. See Figure 1 for an illustration of the algorithm to construct T.

Consider some $x_j, x_i \in X$, if the edge $e = \{x_j, x_i\} \in E(T)$, then clearly this pair has distortion 1. Otherwise, let P be the unique path between x_j and x_i in T. Since e is not in T, it is the heaviest edge on the cycle $P \cup \{e\}$, and for any edge $e' \in P$ we

[2]The map f is noncontractive if for any $u, v \in X$, $d_X(u, v) \le d_Y(f(u), f(v))$.

have that $w(e') \le w(e) = \alpha(j) \cdot d(x_j, x_i)$. Consider some $x_k \in X$, and note that there can be at most 2 edges touching x_k in P. If $e' \in P$ is such an edge, and its weight by w was changed by a factor of $\alpha(k)$, then $\alpha(k) \cdot \ell(e') \le \alpha(j) \cdot d(x_j, x_i)$. Summing this over all the possible values of k we obtain that the length of P is at most

$$\sum_{e' \in P} \ell(e') \le 2 \sum_{k=1}^{n} \frac{\alpha(j)}{\alpha(k)} \cdot d(x_j, x_i) \le 2\alpha(j) \cdot d(x_j, x_i) \ . \tag{1}$$

∎

Corollary 1. *For any finite metric space (X, d) and any fixed $0 < \epsilon < 1/2$, there is a (noncontractive) embedding of X into a single tree with priority distortion $O(j(\log j)^{1+\epsilon})$. Furthermore, the distortion of the pairs containing x_1 is only $1 + 3\epsilon$.*

Proof. Take the function $\alpha : \mathbb{N} \to \mathbb{R}$ defined by $\alpha(1) = 1+\epsilon$, and for $j \ge 2$, $\alpha(j) = \frac{j(\ln j)^{1+\epsilon}}{c}$ (c is a constant to be determined later). Then $\sum_{j \ge 3} \frac{1}{\alpha(j)} \le \int_2^\infty \frac{c}{x(\ln x)^{1+\epsilon}} dx = \frac{-c}{\epsilon \cdot \ln^\epsilon x} |_2^\infty = \frac{c}{\epsilon \cdot \ln^\epsilon 2}$. In particular, $\sum_{j \ge 1} \frac{1}{\alpha(j)} = \frac{1}{1+\epsilon} + \frac{c}{2(\ln 2)^{1+\epsilon}} + \frac{c}{\epsilon \cdot \ln^\epsilon 2} \le 1$ for $c = O(\epsilon^2)$. We conclude that $\alpha \in \Phi$. The corollary now follows by Theorem 1, except that it only provides distortion $2(1+\epsilon)$ for pairs containing x_1. To see the improved distortion for pairs (x_1, x_i), consider the proof of Theorem 1. Observe that in the case $\{x_1, x_i\} \notin T$, the first edge of the path P from x_1 to x_i has weight at most $d(x_1, x_i)$, while none of the other edges on P are touching x_1. Furthermore, since $1/\alpha(1) > 1 - \epsilon$, we have that $\sum_{k=2}^{\infty} 1/\alpha(k) < \epsilon$, and so we can replace (1) by

$$\sum_{e' \in P} \ell(e') \le d(x_1, x_i) + 2 \sum_{k=2}^{n} \frac{\alpha(1)}{\alpha(k)} \cdot d(x_1, x_i) \le (1 + 3\epsilon) \cdot d(x_1, x_i) \ .$$

∎

3.2. Lower bound. Here we show a matching lower bound (up to a constant), which is only 2 for trees without Steiner nodes[3] on the possible functions admitting an embedding into a tree with priority distortion. We first show that a (nondecreasing) function which is not in Φ cannot bound the priority distortion in a spanning tree embedding. Then using an argument similar to that of [Gup01], we extend this for arbitrary dominating trees,[4] while losing a factor of 8 in the lower bound.

Theorem 2. *For any nondecreasing function $\alpha : \mathbb{N} \to \mathbb{R}$ with $\alpha \notin \Phi$, there exists an integer n, a graph $G = (V, E)$ with $|V| = n$ vertices, and a priority ranking of V, such that no spanning tree of G has priority distortion strictly less than α.*

Proof. Since $\alpha \notin \Phi$, there exists an integer n' such that $\sum_{i=1}^{n'} 1/\alpha(i) > 1$. Take some integer $n > n'$ such that $\frac{n}{\alpha(i)+1}$ is an integer for all $1 \le i \le n'$ (assume without loss of generality (w.l.o.g.) that the $\alpha(i)$ are rational numbers). Then let $G = C_n$, a cycle on n points with unit weight on the edges. Clearly, a spanning tree of C_n is obtained by removing a single edge, thus we will choose the priorities $x_1, \dots, x_n \in V$ in such a way that no edge can be spared.

Seeking contradiction, assume that there exists a spanning tree with priority distortion less than α. Let x_1 be an arbitrary vertex, and note that if u is a vertex within distance (in G) $a_1 = \frac{n}{\alpha(1)+1}$ from x_1, then all the edges on the shortest path from x_1 to u must remain in the tree. Otherwise, the distortion of the pair $\{x_1, u\}$ will be at least $\frac{n-a_1}{a_1} = \alpha(1)$. There are $\frac{2n}{\alpha(1)+1}$ such edges that must belong to the

[3]We say that the target tree has Steiner nodes if it contains more vertices than the original graph.
[4]A tree T dominates a graph G if $d_T \ge d_G$.

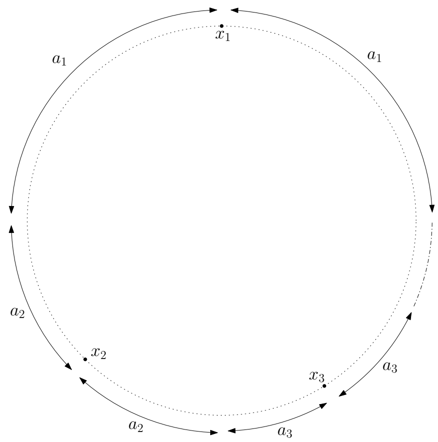

FIG. 2. *An illustration for the proof of Theorem 2. As all the pairs containing x_i cannot suffer distortion greater than or equal to $\alpha(i)$, all the edges of distance at most a_i from x_i cannot be deleted from the tree. As $\sum a_i > n$, placing $x_1, x_2, \ldots$ so that the relevant sets of edges are disjoint and cover all the edges, there is no edge that can be deleted.*

tree (since we consider vertices from both sides of x_1). Now take x_2 to be a vertex at distance $\frac{n}{\alpha(1)+1} + \frac{n}{\alpha(2)+1}$ from x_1. By a similar argument, the $\frac{2n}{\alpha(2)+1}$ edges closest to x_2 must be in the tree as well. Observe that these edges form a continuous sequence on the cycle with those edges near x_1. Continue in this manner to define $x_3, \ldots, x_{n'}$, and conclude that there are at least

$$\sum_{i=1}^{n'} \frac{2n}{\alpha(i)+1} \geq \sum_{i=1}^{n'} \frac{n}{\alpha(i)} > n \tag{2}$$

edges that are not allowed to be removed, but this is a contradiction, as there are only n edges in C_n. See Figure 2 for an illustration of this argument. ∎

THEOREM 3. *For any nondecreasing function $\alpha : \mathbb{N} \to \mathbb{R}$ with $\alpha \notin \Phi$, there exists an integer n, a metric (X, d) on n points, and a priority ranking $x_1, \ldots, x_n \in X$, such that there is no embedding of X into a dominating tree metric with priority distortion strictly less than $\alpha/8$.*

Proof. Take n, the metric (X, d) induced by C_n, and the same priority ranking as in Theorem 2. First consider any tree T with exactly n vertices, but which is not necessarily spanning. That is, T is allowed to have edges that did not exist in C_n. Since T must be dominating, we may assume that an edge in T connecting vertices of distance k in C_n will have weight exactly k (if it has larger weight, reducing it to k can only improve the distortion). We extend an argument of [Gup01] to prove that the priority distortion of T is at least α.

The argument in section 7 of [Gup01] says that T can be replaced by a tree T' satisfying $d \leq d_{T'} \leq d_T$, and such that any vertex in T' has at most one edge to its left semicircle and one edge to its right semicircle.[5] A crucial observation (made in [Gup01]) is that for any pair of vertices at distance k in C_n, their distance in T' can be either k or at least $n - k$. Now we may use similar reasoning as in the proof of

[5]If the vertices of C_n are labeled $0, 1, \ldots, n-1$ as ordered on the cycle, the right semicircle of vertex i is $\{i+1, i+2, \ldots i + \lfloor n/2 \rfloor\}$ (addition is modulo n), and the left semicircle is $V \setminus \{i, i+1, i+2, \ldots i + \lfloor n/2 \rfloor\}$.

Theorem 2; assume that x_1 is the ith vertex of C_n, and observe that any vertex $i+j$ for $1 \leq j \leq a_1$, must be connected by an edge to one of the vertices $i, i+1, \ldots, i+j-1$, as otherwise $d_{T'}(i, i+j) \geq n - a_1$, and the distortion of the pair $\{x_1, j\}$ will be at least $\alpha(1)$. Notice that the edges x_2 forced to exist are disjoint from those of x_1. It follows that for each $1 \leq i \leq n'$, x_i forces at least $\frac{2n}{\alpha(i)+1}$ disjoint edges to be in the tree, which is impossible due to (2).

Finally, consider arbitrary dominating tree metrics, which may have Steiner nodes (nodes which no vertex of C_n is mapped onto). By a result of [Gup01], such nodes may be removed while increasing the distance between any pair of points by at most 8, so we conclude that such a tree cannot have priority distortion strictly less than $\alpha/8$. ∎

4. Probabilistic embedding into ultrametrics with prioritized distortion. In this section, we present our probabilistic embedding into trees with prioritized expected distortion. Specifically, we generalize the embedding of [FRT04] which has a worst-case expected distortion guarantee, to prioritize expected distortion.

THEOREM 4. *For any metric space (X, d), there exists a distribution over embeddings of X into ultrametrics with expected prioritized distortion $O(\log j)$.*

Proof. Let $x_1, \ldots, x_n$ be the priority ranking of X, and let Δ be the diameter of X. We assume w.l.o.g. that the minimal distance in X is 1, and let δ be the minimal integer so that $\Delta \leq 2^\delta$. We shall create a hierarchical laminar partition, where for each $i \in \{0, 1, \ldots, \delta\}$, the clusters of level i have diameter at most 2^i, and each of them is contained in some level $i+1$ cluster. The ultrametric is built in the natural manner, the root corresponds to the level δ cluster which is X, and each cluster in level i corresponds to an inner node of the ultrametric with label 2^i, whose children correspond to the level $i-1$ clusters contained in it. The leaves correspond to singletons, that is, to the elements of X. Clearly, the ultrametric will dominate (X, d).

In order to define the partition, we choose a random permutation $\pi : X \to [n]$ which is strongly correlated with the priority ranking, and in addition we choose a random number $\beta \in [1, 2]$ from an appropriate distribution. (See line 2 of Algorithm 1.) Let $K_0 = \{x_1, x_2\}$, and for any integer $1 \leq j \leq \lceil \log \log n \rceil$ let $K_j = \{x_h : 2^{2^{j-1}} < h \leq 2^{2^j}\}$ The permutation π is created by choosing a uniformly random permutation on each K_i, and concatenating these. Note that $\pi^{-1}(\{h \in \mathbb{N} \ : \ h \in (2^{2^{j-1}}, 2^{2^j}]\}) = K_j$, and $\pi^{-1}(\{1, 2\}) = K_0$.

In each step i, we partition a cluster S of level $i+1$ as follows. Each point $x \in S$ chooses the point $u \in X$ with minimal value according to π among the points of distance at most $\beta_i := \beta \cdot 2^{i-2}$ from x, and joins to the cluster of u. Observe that $x \in S$ might belong to the cluster of u where $u \notin S$. In particular, a point may not belong to the cluster associated with it, and some clusters may be empty (which we can discard). The description of the hierarchical partition appears in Algorithm 1.

Let T denote the ultrametric created by the hierarchical partition of Algorithm 1, and $d_T(u, v)$ the distance between u to v in T. Consider the clustering step at some level i, where clusters in D_{i+1} are picked for partitioning. In each iteration l, all unassigned points z such that $d(z, \pi(l)) \leq \beta_i$ assign themselves to the cluster of $\pi(l)$. Fix an arbitrary pair $\{v, u\}$. We say that center w *settles* the pair $\{v, u\}$ at level i, if it is the first center so that at least one of u and v gets assigned to its cluster. Note that exactly one center w settles any pair $\{v, u\}$ at any particular level. Further, we say that a center w *cuts* the pair $\{v, u\}$ at level i, if it settles them at this level, and exactly one of u and v is assigned to the cluster of w at level i. Whenever w cuts a pair $\{v, u\}$ at level i, $d_T(v, u)$ is set to be $2^{i+1} \leq 8\beta_i$. We charge this length to the point w and define $d_T^w(v, u)$ to be $\sum_i \mathbf{1}(w \text{ cuts } \{v, u\} \text{ at level } i) \cdot 8\beta_i$ (where $\mathbf{1}(\cdot)$ denotes an indicator function). We also define $d_T^{K_j}(v, u) = \sum_{w \in K_j} d_T^w(v, u)$. Clearly, $d_T(v, u) \leq \sum_j d_T^{K_j}(v, u)$.

Algorithm 1 `Modified FRT`(X, π).

Choose a random permutation $\pi : X \to [n]$ as above.
Choose $\beta \in [1, 2]$ randomly by the distribution with the following probability density function $p(x) = \frac{1}{x \ln 2}$.
Let $D_\delta = X$; $i \leftarrow \delta - 1$.
while D_{i+1} has nonsingleton clusters **do**
    Set $\beta_i \leftarrow \beta \cdot 2^{i-2}$.
    for $l = 1, \ldots, n$ **do**
        for every cluster S in D_{i+1} **do**
            Create a new cluster in D_i, consisting of all unassigned points in S closer than β_i to $\pi(l)$.
        end for
    end for
    $i \leftarrow i - 1$.
end while

Fix some $0 \leq j \leq \lceil \log \log n \rceil$. Our next goal is to bound the expected value of $d_T^{K_j}(v, u)$ by $O(\log(|K_j|))$. We arrange the points of K_j in nondecreasing order of their distance from the pair $\{v, u\}$ (breaking ties arbitrarily). Consider the sth point w_s in this sequence. W.l.o.g. assume that $d(w_s, v) \leq d(w_s, u)$. For a center w_s to cut $\{v, u\}$, it must be the case that

1. $d(w_s, v) \leq \beta_i < d(w_s, u)$ for some i;
2. w_s settles $\{v, u\}$ at level i.

Note that for each $x \in [d(w_s, v), d(w_s, u))$, the probability that $\beta_i \in [x, x + dx)$ is at most $\frac{dx}{x \cdot \ln 2}$. Conditioning on β_i taking such a value x, any one of $w_1, \ldots, w_s$ can settle $\{v, u\}$. The probability that w_s is the first in the permutation π among $w_1, \ldots w_s$ is $\frac{1}{s}$. (In fact, there may be points from $\bigcup_{0 \leq r < j} K_r$ that settle $\{v, u\}$ before w_s. It is safe to ignore that, as it can only decrease the probability that w_s cuts $\{v, u\}$.) Thus, we obtain

$$\text{(3)}\quad \mathbb{E}[d_T^{w_s}(v, u)] \leq \int_{d(w_s,v)}^{d(w_s,u)} 8x \cdot \frac{dx}{x \ln 2} \cdot \frac{1}{s} = \frac{8}{s \cdot \ln 2}(d(w_s, u) - d(w_s, v)) \leq \frac{16}{s} \cdot d(v, u) .$$

Hence, we conclude

$$\text{(4)}\qquad \mathbb{E}[d_T^{K_j}(v, u)] \leq \sum_{w_s \in K_j} \mathbb{E}[d_T^{w_s}(v, u)] \overset{(3)}{\leq} 16 d(v, u) \sum_{s=1}^{|K_j|} \frac{1}{s} = \log |K_j| \cdot O(d(v, u)) .$$

Assume $v = x_h$ is the hth vertex in the priority ranking for some $h > 2$. Let a be the integer such that $v \in K_a$, and recall that $2^{2^{a-1}} < h \leq 2^{2^a}$, i.e., $2^a \leq 2 \log h$. The crucial observation is that if $y \in K_b$ such that $b > a$, then y cannot settle $\{v, u\}$. The reason is that v always appears before y in π, so v will surely be assigned to a cluster when it is the turn of y to create a cluster. This leads to the conclusion that for all

$b > a$, $\mathbb{E}[d_T^{K_b}(v,u)] = 0$. We conclude

$$\begin{aligned}\mathbb{E}[d_T(v,u)] &\le \sum_{j=0}^{a} \mathbb{E}[d_T^{k_j}(v,u)] \\ &\overset{(4)}{\le} O(d(v,u)) \sum_{j=0}^{a} \log |K_j| \\ &= O(d(v,u)) \sum_{j=0}^{a} \log \left(2^{2^j}\right) \\ &= O(d(v,u)) \sum_{j=0}^{a} 2^j \\ &= O(d(v,u)) \cdot 2^a \\ &= O(d(v,u)) \cdot \log h \ .\end{aligned}$$

When $h \in \{1,2\}$ we can take $a = 0$, and thus obtain a bound of $O(d(v,u))$. ∎

5. Distance oracles with prioritized stretch. In this section we consider distance oracles where the stretch scales with the priority of the vertices. See section 2 for the basic definitions. A classical result of [TZ05], with improved query time and size due to [Che14, Che15], asserts that for any parameter $t \ge 1$ and any graph on n vertices, there exists a $(2t-1)$-stretch distance oracle of space $O(n^{1+1/t})$ with $O(1)$ query time.

5.1. Prioritized stretch with small space. Our first result provides a range of distance oracles with prioritized stretch and extremely low space. They also exhibit a somewhat nonintuitive (although very good) dependence of the stretch on the priority of the vertices. The drawbacks of these oracles are that they cannot report the approximate paths in the graph between the queried vertices, and it is not clear if they can be distributed as a labeling scheme.

For the sake of brevity, denote $\tau(j) = \left\lfloor \frac{\log n}{\log(n/j)} \right\rfloor$ (where n is always the number of vertices). For a function $f : \mathbb{N} \to \mathbb{N}$, define its iterative application $F : \mathbb{N} \to \mathbb{N}$ as follows: $F(0) = 1$, and, for integer $k \ge 1$, as $F(k) = f(F(k-1))$. That is, $F(k)$ is determined by iteratively applying f for k times starting at 1.

THEOREM 5. *Let $G = (V,E)$ be a weighted graph on n vertices. For any positive integer T, let $f : \mathbb{N} \to \mathbb{R}_+$ be any monotone increasing function such that $f(1) = 2$ and $F(T) \ge \log n$. Then there exists a distance oracle that requires space $O(T \cdot n)$, has query time $O(1)$, and prioritized stretch*

$$\min \{4f(\tau(j)) - 5, \log n\} \ .$$

COROLLARY 2. *Any weighted graph $G = (V,E)$ on n vertices admits distance oracles with the following possible trade-offs between space and prioritized stretch:*

(1) *space $O(n \log n)$ and prioritized stretch* $\min\{4\tau(j) - 1, \log n\}$*;*
(2) *space $O(n \log\log n)$ and prioritized stretch* $\min\{8\tau(j) - 5, \log n\}$*;*
(3) *space $O(n \log\log\log n)$ and prioritized stretch* $\min\{4\tau(j)^2 - 5, \log n\}$*;*
(4) *space $O(n \log^* n)$ and prioritized stretch* $\min\{4 \cdot 2^{\tau(j)} - 5, \log n\}$.

Observe that the first two oracles have stretch 3 for all points of priority rank less than $\sqrt{n}$, and that in all of these oracles, for any fixed $\epsilon > 0$, all vertices of priority at most $n^{1-\epsilon}$ have *constant stretch*.

Proof of Corollary 2. All the trade-offs follow by simple choices for T and f, which are described in the next bullets.

- For the first trade–off let $T = \log n$ (assume w.l.o.g. this is an integer), and take the function $f(k) = k + 1$, so that $F(k) = k + 1$ as well for all k, so indeed $F(T) \geq \log n$. Thus the space is indeed $O(n \log n)$, and the prioritized stretch is $\min\{4\tau(j) - 1, \log n\}$ by Theorem 5.
- For the second trade-off, using $T = \log\log n$, it suffices to take $f(k) = 2k$, so that $F(k) = 2^k$ and $F(T) = \log n$ as required. The space is now $O(n \log\log n)$ and the prioritized stretch is as promised applying Theorem 5 again.
- In the third trade-off we use $T = 1 + \log\log\log n$, and let $f(1) = 2$ and for $k \geq 2$, $f(k) = k^2$. It implies that $F(k) = 2^{2^{k-1}}$. The bounds on the space and the prioritized stretch follow as before.
- The final trade-off holds by taking $T = \log^* n - 1$, and setting $f(k) = 2^k$, so that $F(k) = \text{tower}(k)$.[6] The bounds on the space and the prioritized stretch follow as before. □

We now turn to proving the theorem, and start with the following lemma.

LEMMA 1. *For any $t \geq 1$ and any graph $G = (V, E)$ on n vertices with a subset $K \subseteq V$ of size $|K| = k$, there exists a distance oracle which can answer in $O(1)$ time queries on every pair in $K \times V$ with stretch $4t - 1$, using space $O(k^{1+1/t} + n)$.*

Proof. Apply the distance oracle of [Che15] on the complete graph $G' = (K, E')$ with parameter t, where the weight of each edge in E' is the shortest path distance in G between its endpoints. This gives stretch $2t - 1$ for any pair in $K \times K$ and requires space $O(k^{1+1/t})$. For every vertex $u \in V \setminus K$, store only $d_G(u, K)$ and the name of the vertex $k_u \in K$ that manifests this distance (that is, $d_G(u, k_u) = d_G(u, K)$). We obtain a data structure of space $O(k^{1+1/t} + n)$. To answer a distance query between $v \in K$ and $u \in V$, report $\tilde{d}(v, k_u) + d_G(k_u, u)$, where $\tilde{d}$ is the distance reported by the oracle of G'. It remains to bound the stretch: observe that since k_u is the closest vertex to u in K, we have that $d_G(v, k_u) \leq d_G(v, u) + d_G(k_u, u) \leq 2d_G(u, v)$, and thus the reported distance is bounded as follows,

$$\tilde{d}(v, k_u) + d_G(k_u, u) \leq (2t - 1)d_G(v, k_u) + d_G(u, v) \leq (4t - 1)d_G(u, v) .$$

Using the triangle inequality and that the reported distance is never larger than the original,

$$\tilde{d}(v, k_u) + d_G(k_u, u) \geq d_G(v, k_u) + d_G(k_u, u) \geq d_G(u, v) . \qquad \square$$

We are finally ready to prove Theorem 5.

Proof of Theorem 5. Let $x_1, \dots, x_n \in V$ be the priority ranking of V. For each $i \in [T]$, let $S_i = \{x_j \ : \ 1 \leq j \leq n^{1-1/F(i)}\}$, and apply the oracle of Lemma 1 on G with the set S_i and parameter $t_i = F(i) - 1$, let O_i be the resulting oracle.[7] Also invoke the oracle O_{MN} of [MN06] on G, that has stretch $\log n$ on all pairs using only $O(n)$ space (with $O(1)$ query time).

Observe that for each $i \in [T]$, the stretch t_i was chosen so that $(1 - 1/F(i)) \cdot (1 + 1/t_i) = 1$, so that the oracle O_i has space

$$O(|S_i|^{1+1/t_i} + n) = O(n) .$$

[6] tower(k) is defined as tower(0) = 1 and $\text{tower}(k) = 2^{\text{tower}(k-1)}$, so that $\text{tower}(\log^* n) = n$.

[7] Since $F(0) = 1$ and f is strictly monotone, it follows that $F(i) \geq 2$ for all $i \geq 1$, so that $t_i \geq 1$.

The total space is thus $O(T \cdot n)$, as promised. It remains to prove the prioritized stretch guarantee. Fix any $v = x_j$, and let i be the minimal such that $x_j \in S_i$ (observe that if $j > n/2$ there is not necessarily any such i). For $i = 1$ the stretch guaranteed by O_1 is $4t_i - 1 = 4(F(1)-1)-1 = 3$, as promised (recall that $f(k) \geq 2$ for all $k \geq 1$, so the required stretch is never smaller than 3). For $i > 1$, by minimality of i it follows that $j > n^{1-1/F(i-1)}$, that is, $F(i-1) \leq \left\lfloor \frac{\log n}{\log(n/j)} \right\rfloor = \tau(j)$ (since $F(i-1)$ is an integer). The stretch of O_i for v with any other point is at most

$$4(F(i)-1)-1 = 4F(i)-5 = 4f(F(i-1))-5 \leq 4f\left(\tau(j)\right)-5\ ,$$

while the stretch of O_{MN} is at most $\log n$ for all pairs, which handles the case no i exists, and allows us to report the minimum of the two terms. The query time is $O(1)$, since each v stores the relevant oracle for it, whose query time is $O(1)$. ∎

5.2. Prioritized distance oracles with bounded prioritized stretch. In this section we prove the following theorem, which prioritizes the stretch of the distance oracle of [TZ05]. Unlike the oracles of Theorem 5, this oracle can also support path queries, that is, return a path in the graph that achieves the required stretch, in time proportional to its length (plus the distance query time). Additionally, it can be distributed as a labeling scheme, which we exploit in the next section. Furthermore, this oracle matches the best known bounds for the worst-case stretch of [TZ05], which are conjectured to be optimal.

THEOREM 6. *Let $G = (V, E)$ be a graph with n vertices. Given a parameter $t \geq 1$, there exists a distance oracle of space $O(tn^{1+1/t})$ with prioritized stretch $2\lceil \frac{t \log j}{\log n} \rceil - 1$ and query time $O(\lceil \frac{t \log j}{\log n} \rceil)$.*

Overview. Recall that in the distance oracle construction of [TZ05], a sequence of sets $V = A_0 \supseteq A_1 \supseteq \cdots \supseteq A_t = \emptyset$ is sampled randomly, by choosing each element of A_{i-1} to be in A_i with probability $n^{-1/t}$. We make the crucial observation that the distance oracle provides improved stretch of $2(t-i)-1$, rather than $2t-1$, to points in A_i. However, as these sets are chosen randomly, they have no correlation with our given priority list over the vertices. We therefore alter the construction, to ensure that points with high priority will surely be chosen to A_i for sufficiently large i.

Proof of Theorem 6. Let $x_1, \ldots, x_n \in V$ be the priority ranking of V. For each $i \in \{0, 1, \ldots, t-1\}$ let $S_i = \{x_j \ : \ 1 \leq j \leq n^{1-i/t}\}$. Let $A_0 = V$, $A_t = \emptyset$, and for each $1 \leq i \leq t-1$ define A'_i by including every element of A_{i-1} with probability $n^{-1/t}/2$, and let $A_i = A'_i \cup S_i$. For each $v \in V$ and $0 \leq i \leq t-1$, define the ith pivot $p_i(v)$ as the nearest point to v in A_i, and $B_i(v) = \{w \in A_i \ : \ d(v,w) < d(v, A_{i+1})\}$.[8] Also the *bunch* of v is defined as $B(v) = \bigcup_{0 \leq i \leq t-1} B_i(v)$. The distance oracle will store in a hash table, for each $v \in V$, all the distances to points in $B(v)$, and also the $p_i(v)$ vertices.

The query algorithm for the distance between u, v is essentially the same as in [TZ05], the main difference is that we start the process at level i rather than level 0, for a specified value of i.

Stretch. Let $v = x_j$ be the jth point in the ordering for some $j > 1$, and fix any $u \in V$. (For $j = 1$, observe that every vertex of A_{t-1} lies in all the bunches, so when considering $x_1 \in A_{t-1}$, we have that $x_1 \in B(u)$ and so Algorithm 2 will return the exact distance.) Let $0 \leq i \leq t-1$ be the integer satisfying that $n^{1-(i+1)/t} <$

[8]We assume that $d(v, \emptyset) = \infty$ (this is needed as $A_t = \emptyset$).

Algorithm 2 `Dist`(v, u, i).

```
w ← v;
while w ∉ B(u) do
    i ← i + 1;
    (u, v) ← (v, u);
    w ← p_i(v);
end while
return d(w, u) + d(w, v);
```

$j \le n^{1-i/t}$, that is, the maximal i such that $v \in S_i$. By definition we have that $v \in A_i$ as well, so we may run `Dist`(v, u, i). Assuming that all operations in the hash table cost $O(1)$, the query time is $O(t-i)$. The stretch analysis is similar to [TZ05]: letting u_k, v_k, and w_k be the values of u, v, and w at the kth iteration, it suffices to show that at every iteration in which the algorithm did not stop, $d(v_k, w_k)$ increases by at most $d(u, v)$. It suffices because there are at most $t-1-i$ iterations (since $w_{t-1} \in A_{t-1}$, it lies in all bunches), so if ℓ is the final iteration, it must be that $d(v_\ell, w_\ell) \le (\ell - i) \cdot d(u, v)$ (initially $d(w_i, v_i) = 0$), and by the triangle inequality $d(w_\ell, u_\ell) \le d(u, v) + d(v_\ell, w_\ell) \le (\ell - i + 1) \cdot d(u, v)$, and as $\ell \le t-1$ we conclude that

$$d(w, u) + d(w, v) \le (2(t-i) - 1) \cdot d(u, v) \ .$$

To see the increase by at most $d(u, v)$ at every iteration, we first note that $w_i = v_i \in A_i$ (this fact enables us to start at level i rather than in level 0). In the kth iteration, observe that as $w_k \notin B(u_k)$ but $w_k \in A_k$, it must be that $d(u_k, p_{k+1}(u_k)) \le d(u_k, w_k)$. The algorithm sets $w_{k+1} = p_{k+1}(u_k)$, $v_{k+1} = u_k$, and $u_{k+1} = v_k$, so we get that

$$\begin{aligned} d(v_{k+1}, w_{k+1}) &= d(u_k, p_{k+1}(u_k)) \le d(u_k, w_k) \le d(u_k, v_k) + d(v_k, w_k) \\ &= d(u, v) + d(v_k, w_k) \ . \end{aligned}$$

Note that as $n^{1-(i+1)/t} < j \le n^{1-i/t}$, it follows that $t - i - 1 < \frac{t \log j}{\log n} \le t - i$, so that $t - i = \lceil \frac{t \log j}{\log n} \rceil$. The guaranteed stretch for pairs containing x_j is thus bounded by $2\lceil \frac{t \log j}{\log n} \rceil - 1$ (or stretch 1 for x_1).

Space. Fix any $u \in V$, and let us analyze the expected size of $B(u)$. Fix any $0 \le i \le t-2$, and consider $B_i(u)$. Assume we have already chosen the set A_i, and arrange the vertices of $A_i = \{a_1, \dots a_m\}$ in order of increasing distance to u. Note that if a_r is the first vertex in the ordering to be in A_{i+1}, then $|B_i(u)| = r - 1$. Every vertex of A_i is either in S_{i+1} and thus will surely be included in A_{i+1}, otherwise it has probability $n^{-1/t}/2$ to be in A'_{i+1} and so in A_{i+1} as well. The number of vertices that we see until the first success (being in A_{i+1}) is stochastically dominated by a geometric distribution with parameter $p = n^{-1/t}/2$, which has expectation $2n^{1/t}$. For the last level $t-1$, note that each vertex in $S_i \setminus S_{i+1}$ has probability exactly $(n^{-1/t}/2)^{t-1-i} = n^{-1+(i+1)/t}/2^{t-1-i}$ to be included in A_{t-1}, independently of all other vertices. As $|S_i \setminus S_{i+1}| \le |S_i| = n^{1-i/t}$, the expected number of vertices in A_{t-1} is

$$\sum_{i=0}^{t-1} n^{1-i/t} \cdot n^{-1+(i+1)/t} / 2^{t-1-i} < 2n^{1/t} \ . \tag{5}$$

This implies that $\mathbb{E}[|B_{t-1}(u)|] \le 2n^{1/t}$ as well, and so $\mathbb{E}[|B(u)|] \le 2t \cdot n^{1/t}$. The total expected size of all bunches is therefore at most $2t \cdot n^{1+1/t}$. ∎

6. Prioritized distance labeling. In this section we discuss distance labeling schemes, in which every vertex receives a short label, and it should be possible to approximately compute the distance between any two vertices from their labels alone. The novelty here is that we would like "important" vertices, those that have high priority, to have both improved stretch and also short labels.

6.1. Distance labeling with prioritized stretch and size. We begin by showing that the stretch-prioritized oracle of Theorem 6 can be made into a labeling scheme, with the same stretch guarantees, and with a small label for high ranking points. The result has some dependence on n in the label size, and it seems to be interesting particularly for large values of t. Indeed, we shall use this result with parameter $t = \log n$ in the following, to obtain a fully prioritized label size which will be independent of n, and can support any desired maximum stretch. Furthermore, this result is the basis for our routing schemes with prioritized label size and stretch.

THEOREM 7. *For any graph $G = (V, E)$ with n vertices and any $t \geq 1$, there exists a distance labeling scheme with prioritized stretch $2\lceil \frac{t \log j}{\log n} \rceil - 1$ and prioritized label size $O(n^{1/t} \cdot \log j)$.*

Proof. Using the same notation as section 5, the label of vertex $v \in V$ consists of its hash table (which contains distances to all points in the bunch $B(v)$, and the identity of the pivots $p_i(v)$ for $0 \leq i \leq t-1$). Note that Algorithm 2 uses only this information to compute the approximate distance. The stretch guarantee is prioritized as above, and it remains to give an appropriate bound on the label sizes.

Let $x_1, \ldots, x_n \in V$ be the priority ranking of V. Fix a point $v = x_j$ for some $j > 1$, and let i be the maximal such that $v \in S_i$. Note that this implies that $t - i - 1 < \frac{t \log j}{\log n}$. Observe that $B_0(v) \cup \cdots \cup B_{i-1}(v) = \emptyset$, so it remains to bound the size of $B_i(v), \ldots, B_{t-1}(v)$. For the last set $B_{t-1}(v) = A_{t-1}$, let $\mathcal{E}$ be the event that $|A_{t-1}| \leq 8n^{1/t}$. We already noted in (5) that the expected size of A_{t-1} is at most $2n^{1/t}$, thus using Markov inequality, with probability at least $3/4$ event $\mathcal{E}$ holds.

For $i \leq k \leq t-2$, let X_k be a random variable distributed geometrically with parameter $p = n^{-1/t}/2$, thus $\mathbb{E}[X_k] = 2n^{1/t}$ for all k. We noted above that the distribution of X_k is stochastically dominating the cardinality of $B_k(v)$, thus it suffices to bound $\sum_{k=i}^{t-2} X_k$. Observe that for any integer s, if $\sum_{k=i}^{t-2} X_k > s$, then it means that in a sequence of s independent coin tosses with probability p for heads, we have seen less than $t-1-i$ heads. That is, if $Z \sim \mathrm{Bin}(s, p)$ is a binomial random variable, then

$$\Pr\left[\sum_{k=i}^{t-2} X_k > s\right] = \Pr[Z < t-1-i] \leq \Pr\left[Z < \frac{t \log j}{\log n}\right] \leq \Pr[Z < \log j] \ .$$

Take $s = 16n^{1/t} \cdot \log j$ (assume this is an integer), so that $\mu := \mathbb{E}[Z] = 8 \log j$, and by a standard Chernoff bound

$$\Pr[Z < \log j] = \Pr[Z < \mu/8] \leq e^{-3\mu/8} < 1/j^3 \ .$$

Let $\mathcal{F}$ be the event that for some $2 \leq j \leq n$, $\left|\bigcup_{k=0}^{t-2} B_k(x_j)\right| > 16n^{1/t} \cdot \log j$. By taking a union bound over all $2 \leq j \leq n$ (note that the bound is nonuniform, and depends on j), we obtain that

$$\Pr[\mathcal{F}] \leq \sum_{j=2}^{n} \Pr\left[\left|\sum_{k=0}^{t-2} B_k(x_j)\right| > 16n^{1/t} \cdot \log j\right] \leq \sum_{j=2}^{n} 1/j^3 < 1/4 \ .$$

We conclude that with probability at least $1/2$ both events $\mathcal{E}$ and $\bar{\mathcal{F}}$ hold, which means that the size of the bunch of each x_j is bounded by $O(n^{1/t} \cdot \log j)$, as required. (Recall that $x_1 \in A_{t-1}$, so its label size is $|A_{t-1}| \leq 8n^{1/t}$ when event $\mathcal{E}$ holds.) ◻

COROLLARY 3. *Any graph $G = (V, E)$ has a distance labeling scheme with prioritized stretch $2\lceil \log j \rceil - 1$ and prioritized label size $O(\log j)$.*

6.2. Distance labeling with prioritized label size. In this section we construct a labeling scheme in which the maximum stretch is fixed for all points, and the label size is fully prioritized and independent of n.

THEOREM 8. *For any graph $G = (V, E)$ and an integer $t \geq 1$, there exists a distance labeling scheme with stretch $2t - 1$ and prioritized label size $O(j^{1/t} \cdot \log j)$.*

Proof overview. The idea is to partition the vertices into $m := \lceil \frac{\log n}{t} \rceil$ sets $S_1, \ldots, S_m$, and to apply the result of section 6.1 in conjunction with a variation of the *source-restricted distance oracles* of [RTZ05], using a labeling scheme rather than an oracle. In a source restricted labeling scheme on X with a subset $S \subseteq X$, only distances between pairs in $S \times X$ can be queried. Replacing the source restricted oracle with a labeling scheme demands that we use an analysis similar to section 6.1 to guarantee a prioritized bound on the label sizes. We will apply this for each $i \in \{2, 3, \ldots, m\}$ with $X = S_i \cup \cdots \cup S_m$ and the subset S_i. Thus an element of S_i will have a label which consists of i schemes, and we will guarantee that their sizes form a geometric progression, so that the total label size is sufficiently small.

As it turns out, the construction of [RTZ05] is inadequate for the first 2^t elements S_1, which have very strict requirement on their label size. We will use the construction of section 6.1 to handle distances involving the elements in S_1. Fortunately, the stretch incurred by this construction is $2\lceil \log j \rceil - 1$ which is bounded by $2t - 1$ for the first 2^t elements in the ranking. We begin by stating the source-restricted distance labeling, based on [RTZ05].

THEOREM 9. *For any integer $t \geq 1$, any graph $G = (V, E)$ and a subset $S \subseteq V$, there exists a source-restricted distance labeling scheme with stretch $2t - 1$ and prioritized label size $O(|S|^{1/t} \cdot \log j)$.*

Proof. The observation made in [RTZ05] is that to obtain a source-restricted distance oracle, it suffices to sample the random sets $S = A_0 \supseteq A_1 \supseteq \cdots \supseteq A_t = \emptyset$ only from S, where each element of A_{i-1} is included in A_i independently with probability $|S|^{-1/t}$. They show that defining the bunches as in [TZ05], the resulting stretch is $2t - 1$ for all pairs in $S \times V$. We shall use a similar analysis as in Theorem 7 to argue that this can be made into a labeling scheme. The expected label size is $O(|S|^{1/t})$, and we can show that with constant probability, every point x_j pays only an additional factor of $O(\log j)$. As the proof is very similar, we leave the details to the reader. ◻

Proof of Theorem 8. Let $S_1 = \{x_j \ : \ 1 \leq j \leq 2^t\}$, and for each $i \in \{2, 3, \ldots, m\}$ let $S_i = \{x_j \ : \ 2^{(i-1)t} < j \leq 2^{it}\}$. We have a separate construction for $i = 1$ and for $i > 1$. For the case $i = 1$, use the labeling scheme of Corollary 3 on $G = (V, E)$. For each $2 \leq i \leq m$, apply Theorem 9 on G and the subset S_i, but append the resulting labels only for vertices in $S_i \cup \cdots \cup S_m$.

Fix any $u, v \in V$, and w.l.o.g. assume that $v \in S_i$ has a higher rank than u. This implies that $u \in S_i \cup \cdots \cup S_m$, thus the source restricted labeling scheme for S_i guarantees stretch at most $2t - 1$ for the pair u, v (and u indeed stored the appropriate label). Note that in the case of $v = x_j \in S_1$, the stretch can be improved to $2\lceil \log j \rceil - 1$ (recall that $\log j \leq t$).

We now turn to bounding the label sizes. First consider $v = x_j \in S_1$, then it must be that $j \leq 2^t$. The label size of v is by Corollary 3 at most $O(\log j)$, and this is the final label of v. For $v = x_j \in S_i$ when $i \geq 2$, the label of v consists of labels created for the sets $S_1, \ldots, S_i$. Notice that $2^{t(i-1)} < j \leq 2^{ti}$, so it holds that $2^i = (2^t \cdot 2^{t(i-1)})^{1/t} < 2j^{1/t}$. By Corollary 3 the label due to S_1 is at most $O(\log j)$, and using Theorem 9 the label size of v is at most

$$O(\log j) + \sum_{k=2}^{i} O(|S_k|^{1/t} \cdot \log j) = O(\log j) \cdot \sum_{k=1}^{i} 2^k = O(2^i \cdot \log j) = O(j^{1/t} \cdot \log j) .$$

∎

6.3. Prioritized distance labeling for graphs with bounded separators.

6.3.1. Exact labeling with prioritized size. In this section we exhibit a prioritized exact distance labeling scheme tailored for graphs that admit a small separator. We say that a graph $G = (V, E)$ admits an s-separator, if for any weight function $w : V \to \mathbb{R}_+$, there exists a set $U \subseteq V$ of size $|U| = s$, such that each connected component C of $G \setminus U$, has $w(C) \leq 2w(V)/3$.[9] It is well known that trees admit a 1-separator, and graphs of treewidth k admit a k-separator.

The basic idea for constructing an exact distance labeling scheme based on separators is to create a hierarchical partition of the graph, each time by applying the separator on each connected component. Then the label of a vertex u consists of all distances to all the vertices in the separators of clusters that contain u. To answer a query between vertices u, v, we return the minimum of $d(u, s)+d(v, s)$ for all separator vertices s that u, v have in common in their labels (this is the exact distance, because at some point a vertex on the shortest path from u to v must be chosen to be in a separator). Since at every iteration the number of vertices in each cluster drops by at least a constant factor, after $O(\log n)$ levels the process is complete, thus the label size is at most $O(s \log n)$.

Our improved label size for vertices of high priority, will be based on the following observation: if the weight function w is an indicator for a set $S \subseteq V$ (that is, if $u \in S$, then $w(u) = 1$, and if $u \in V \setminus S$ then $w(u) = 0$), then after $\lceil \log |S| \rceil + 1$ iterations, all vertices of S must have been removed from the graph.

THEOREM 10. *Let $G = (V, E)$ be a graph admitting an s-separator, and let $V = (x_1, \ldots, x_n)$ be a priority ranking of the vertices. Then there exists an exact distance labeling scheme with prioritized label size $O(s \cdot \log j)$.*

Proof. let $S_0 = \{x_1, x_2\}$, and for $1 \leq i \leq \lceil \log \log n \rceil$ let $S_i = \{x_j : 2^{2^{i-1}} < j \leq 2^{2^i}\}$ The hierarchical partition will be performed in $\log \log n$ phases. The ith phase consists of $2^i + 1$ levels. In each level of the ith phase, we generate an s-separator for each remaining connected component C with the following weight function

$$w(u) = \begin{cases} 1 & \text{if } u \in S_i \cap C, \\ 0 & \text{otherwise.} \end{cases}$$

Then this separator is removed from the component. By the observation made above, after at most $1+\log |S_i| \leq 2^i+1$ levels, all remaining components have no vertices from S_i. The label of a vertex $u \in V$ will be the distances to all points in the separators created for components containing u.

[9] For a set $C \subseteq V$, its weight is defined as $w(C) = \sum_{u \in C} w(u)$.

Fix some vertex x_j (for $j > 1$), and assume $x_j \in S_i$. Notice that $2^{i-1} < \log j$. Then the label size of x_j is at most

$$\sum_{k=0}^{i} s \cdot (2^k + 1) = O(s \cdot 2^i) = O(s \cdot \log j) \ .$$

∎

6.3.2. Planar graphs and graphs excluding a fixed minor. While exact distance labeling for planar graphs requires polynomial label size or query time, there is a $1 + \epsilon$ stretch labeling scheme for planar graphs with label size $O(\log n)$ [Tho01, Kle02], which was extended to graphs excluding a fixed minor [AG06]. All these constructions are based on *path separators*: a constant number of shortest paths in the graph, whose removal induces pieces of bounded weight. The label of a vertex consists of distances to carefully selected vertices on these paths. We may use the same methodology as above; generate these path separators for the sets S_i in order, and obtain the following.

THEOREM 11. *Let $G = (V, E)$ be a graph excluding some fixed minor, and $V = (x_1, \ldots, x_n)$ a priority ranking of the vertices. Then for any $\epsilon > 0$ there exists a distance labeling scheme with stretch $1 + \epsilon$ and prioritized label size $O((\log j)/\epsilon)$.*

7. Routing.

7.1. Routing in trees with prioritized labels. In this section we extend a result of [TZ01], and show a routing scheme on trees. The setting is that each vertex stores a routing table, and when a routing request arrives for vertex v, it contains $L(v)$, the label of vertex v. We will show the following.

THEOREM 12. *For any tree $T = (V, E)$ there is a routing scheme with routing tables of size $O(1)$ and labels of prioritized size $\log j + 2 \log\log j + 4$.*

Proof. The proof follows closely the one of [TZ01], with the major difference being the assignments of weights, which gives preference to the high priority vertices, thus ensuring that when routing from the root of the tree to a vertex of rank j, there are $\approx \log j$ junctions that require routing information from the label of the vertex.

Let $x_1, \ldots, x_n$ be the priority ranking of V. Let $S_0 = \{x_1\}$ and for each $1 \le i \le \log n$, let $S_i = \{x_j \ : \ 2^{i-1} < j \le 2^i\}$. Fix an arbitrary root r of the tree T. For every $v \in S_i$ define $p(v) = \frac{1}{2^i \cdot (i+1)^2}$. Note that as $|S_i| \le 2^i$ we have that

$$\sum_{v \in V} p(v) \le \sum_{i=0}^{\log n} \frac{2^i}{2^i \cdot (i+1)^2} \le 2 \ .$$

For each $v \in V$, define the weight of v as $s_v = \sum_{u \in T_v} p(u)$, where T_v is the subtree rooted at v (including v itself). A child v' of v is called *heavy* if its weight is greater than $s_v/2$; otherwise it is called *light*. The root r of the tree will always be considered heavy. Observe that any vertex can have at most one heavy child. The *light level* $\ell(v)$ of a vertex v is defined as the number of light vertices on the path from the root to v, denoted by $Path(v) = (r = v_0, v_1, \ldots, v_k = v)$. The label size of v will be $\ell(v)$ words.

We enumerate all vertices T in depth-first search (DFS) order, where all the light children of a vertex are visited before its heavy child is visited. (The order is otherwise arbitrary.) We identify each vertex v with its DFS number. Let f_v denote the largest descendant of v. Also, let h_v denote its heavy child, if exists. If it does not exist define $h_v = f_v + 1$. Also, let $P(\pi(v))$ denote the port number of the edge connecting v to its parent $\pi(v)$, and $P(h_v)$ denote the port number connecting v to its heavy child (if it exists). The routing table stored at v is $(v, f_v, h_v, P(\pi(v)), P(h_v))$. It requires $O(1)$ words.

Each time an edge from a vertex to one of its light children is taken, the weight of the corresponding subtree decreases by at least a factor of 2. Note that a vertex $v = x_j \in S_i$ has weight at least $w(v) \geq p(v) = \frac{1}{2^i \cdot (i+1)^2}$, and since the root has weight at most 2, it follows that $\ell(v) \leq \log(2 \cdot 2^i \cdot (i+1)^2) = i + 2\log(i+1) + 1$. Since $2^{i-1} < j$, we conclude that

$$\ell(v) \leq \log j + 2\log(\log(j) + 2) + 2 \ .$$

For each index q, $1 \leq q \leq \ell(v)$, denote by i_q the index of the qth light vertex of $Path(v)$. Let $L(v) = (v, (port(v_{i_1-1}, v_{i_1}), \ldots, port(v_{i_{\ell(v)}-1}, v_{i_{\ell(v)}})))$ be the label of v, which consists of its name, and a sequence of at most $\ell(v)$ words containing the port numbers corresponding to the edges leading to light children on $Path(v)$.

The routing algorithm works as follows. Suppose we need to route a message with the header $L(v)$ at a vertex w. The vertex w checks if $w = v$. If it is the case then we are done. Otherwise, w checks if $v \in [w, w+1, \ldots, f_w]$. If it is not the case, then v is not in the subtree of w, and then w sends the message to its parent. Otherwise w checks if $v \in [h_w, h_w + 1, \ldots, f_w]$. If it is the case then the message is sent to the heavy child. Otherwise v is a descendant of a light child of w. The vertex w finds itself in the sequence of $L(v)$, and determines to which light child of w the message should be sent. Then it sends the message to this child. ∎

7.2. Routing in general graphs. To obtain a routing scheme for general graphs, we use the same method as [TZ01], but replace their distance labeling with our prioritized ones from Theorem 7. This routing scheme has the following property: after an initial calculation using the entire label of the destination vertex v, all routing decisions are based on a much shorter *header* appended to the message. In particular, we obtain the following theorem.

THEOREM 13. *For any graph $G = (V, E)$ with priority ranking $x_1, \ldots, x_n$ of V, and any parameter $t \geq 1$, there exists a routing scheme, such that the label size of x_j is at most $\log j \cdot \lceil \frac{t \log j}{\log n} \rceil \cdot (1 + o(1))$, and it stores a routing table of size $O(n^{1/t} \cdot \log j)$. Routing from any vertex into x_j will have stretch at most $4\lceil \frac{t \log j}{\log n} \rceil - 3$ using a header of size $\log j \cdot (1 + o(1))$, while routing from x_j towards any other vertex incurs stretch at most $4\lceil \frac{t \log j}{\log n} \rceil - 1$ using a header of size at most $\log n \cdot (1 + o(1))$.*

Sketch. We use the definitions of section 5.2. Consider the distance labeling scheme given in Theorem 7. Following [TZ05], this labeling scheme yields a tree cover: a collection of subtrees such that vertex $v = x_j$ belongs to at most $|B(v)|$ trees. The tree T_z for vertex z contains z as the root, and the shortest path to all the vertices in $C(z) = \{x \in V \ : \ z \in B(x)\}$. To route from some vertex $u \in V$ to v, it suffices to find an appropriate $z \in B(u) \cap B(v)$, and route in T_z by applying Theorem 12.

The *routing table* stored at each vertex $v \in V$ contains the hash table for its bunch $B(v)$, and the routing table needed to route in T_z for each $z \in B(v)$. Recall that by Theorem 7, $|B(v)| \leq O(n^{1/t} \cdot \log j)$ (where $v = x_j$), and by Theorem 12, the routing table of each tree is of constant size. Assume first that we route towards a high ranked vertex, and let i be the minimal such that $v = x_j \in S_i$. The *label* of v is $((p_i(v), L_i(v)), \ldots, (p_{t-1}(v), L_{t-1}(v)))$, where $L_h(v)$ is the label of v that is required to route in $T_{p_h(v)}$. Note that the label is of size $(t-i)\log j \cdot (1+o(1)) = \log j \cdot \lceil \frac{t\log j}{\log n} \rceil \cdot (1+o(1))$ (the equality follows from a calculation done in section 5.2).

Finding the tree which guarantees the prioritized stretch as in Theorem 7 could have been achieved by using Algorithm 2; alas, this requires knowledge of the bunches of both vertices u and v. It remains to see that using only the label of v and the routing table at u, one can find a tree in the cover which has stretch at most $4\lceil \frac{t\log j}{\log n} \rceil - 3$ for u, v (routing in the tree does not increase the stretch). To see this, let $i \le h \le t-1$ be the minimal such that $p_h(v) \in B(u)$. Following [TZ01], we prove by induction that for each $i \le k \le h$ it holds that

$$d(v, p_k(v)) \le 2(k-i) \cdot d(u,v).$$

The base case for $k = i$ holds as $v = p_i(v)$, assume for k, and for $k+1$: Since $k < h$ it follows that $p_k(v) \notin B(u)$, thus it must be that $d(u, p_{k+1}(u)) \le d(u, p_k(v))$. Now,

$$\begin{aligned} d(v, p_{k+1}(v)) &\le d(v, p_{k+1}(u)) \\ &\le d(v,u) + d(u, p_{k+1}(u)) \\ &\le d(v,u) + d(u, p_k(v)) \\ &\le 2d(v,u) + d(v, p_k(v)) \\ &\le (2(k-i)+2) \cdot d(u,v)\ , \end{aligned}$$

where the last inequality uses the induction hypothesis. Finally, routing through the shortest path tree rooted at $p_h(v)$ will have stretch at most

$$\begin{aligned} d(u, p_h(v)) + d(p_h(v), v) &\le d(u,v) + 2d(v, p_h(v)) \\ &\le (4(h-i)+1) \cdot d(u,v) \\ &\le (4(t-i)-3) \cdot d(u,v) \\ &= \left(4\left\lceil \frac{t\log j}{\log n} \right\rceil - 3\right) \cdot d(u,v)\ , \end{aligned}$$

using that $h \le t-1$ and that $t - i = \lceil \frac{t\log j}{\log n} \rceil$. Note that once the vertex $p_h(v)$ is found, all other vertices on the route from u to v only require the information $(p_h(v), L_h(v))$, which is appended to the message as a header of size $\log j \cdot (1+o(1))$.

We now turn to the case where u is the high ranked vertex, and let i be the minimal index such that $u \in S_i$. Since $u \in A_i$ by definition, we have that $d(v, p_i(v)) \le d(v,u)$. The label of v contains $((p_i(v), L_i(v)), \ldots, (p_{t-1}(v), L_{t-1}(v)))$ (since v has worse rank than u), so we can use the same algorithm as above: find the minimal $i \le h \le t-1$ such that $p_h(v) \in B(u)$, and route in $T_{p_h(v)}$. We can prove by induction that for $i \le k \le h$,

$$d(v, p_k(v)) \le (2(k-i)+1) \cdot d(u,v).$$

The base case $k = i$ holds since we have $d(v, p_i(v)) \le d(u,v)$. The rest of the proof is similar to the one above, and we leave the details to the reader. The final stretch will be $4\lceil \frac{t\log j}{\log n} \rceil - 1$ (the $+1$ will increase it by an additive 2), as required. ∎

COROLLARY 4. *Any graph $G = (V,E)$ with a priority ranking $x_1, \ldots, x_n$ has a fully prioritized routing scheme, such that the label size of x_j is at most $\log^2 j \cdot (1+o(1))$, and it stores a routing table of size $O(\log j)$. Routing from or towards x_j will have stretch at most $4\lceil \log j \rceil - 1$.*

8. Prioritized embedding into normed spaces. We start by providing some notations used in this section. For $p \in [1, \infty]$ and $m \in \mathbb{N}$, $\ell_p^m = (\mathbb{R}^m, \|\cdot\|_p)$ denotes the m-dimensional real vector space with the ℓ_p-norm. Specifically, for $x = (x_1, \ldots, x_m) \in \mathbb{R}^m$ we have $\|x\|_p = (\sum_{i=1}^m |x_i|^p)^{\frac{1}{p}}$. As usual, the ℓ_p-norm induces a metric on $\mathbb{R}^m$, where the distance between $x, y \in \mathbb{R}^m$ is $\|x-y\|_p$. Distortion, priority, and prioritized distortion are defined naturally using this metric. Given some metric space (X, d_X) and two functions $f_1 : X \to \ell_p^{d_1}$ and $f_2 : X \to \ell_p^{d_2}$, their concatenation is a function from X into $\ell_p^{d_1+d_2}$, denoted $f_1 \oplus f_2$.

For a metric space (K, d_K), an embedding $f : K \to \mathbb{R}^m$ is called a (normalized) *Fréchet* embedding if there are some m sets $A_1, \ldots, A_m \subseteq K$ such that f is defined as $f(x) = m^{-1/p} \bigoplus_{i=1}^m d_K(x, A_i)$. A useful property of Fréchet embeddings is that they can be extended into nonexpansive embedding. Formally, suppose (X, d_X) is a metric space, and $K \subseteq X$ admits a Fréchet embedding $f : K \to \ell_p^m$ (with the induced metric). An extension $\hat{f}$ is a function $\hat{f} : X \to \ell_p^m$, such that for every $x \in K$, $f(x) = \hat{f}(x)$. To get a nonexpansive extension for $y \in X$, simply define $\hat{f}(y) = m^{-1/p} \bigoplus_{i=1}^m d_X(y, A_i)$. It is straightforward that $\hat{f}$ is an extension of f. As for every $x, y, \in X$,

$$\left\|\hat{f}(x) - \hat{f}(y)\right\|_p = \left(\sum_{i=1}^m \left|m^{-\frac{1}{p}} \cdot d_X(x, A_i) - m^{-\frac{1}{p}} \cdot d_X(y, A_i)\right|^p\right)^{\frac{1}{p}}$$
$$\leq \left(\frac{1}{m} \cdot \sum_{i=1}^m |d_X(x, y)|^p\right)^{\frac{1}{p}} = d(x, y) \ ,$$

so $\hat{f}$ is also nonexpansive.

8.1. Embedding with prioritized distortion. In this section we study embedding arbitrary metrics into normed spaces, where the distortion is prioritized according to the given ranking of the points in the metric. Our main result is the following

THEOREM 14. *For any $p \in [1, \infty]$, $\epsilon > 0$, and any finite metric space (X, d) with priority ranking $X = (x_1, \ldots, x_n)$, there exists an embedding of X into $\ell_p^{O(\log^2 n)}$ with priority distortion $O(\log j \cdot (\log \log j)^{(1+\epsilon)/2})$.*

Proof overview. Our improved distortion guarantee for high ranked points comes from a variation of Bourgain's embedding [Bou85] of finite metric spaces into ℓ_p space. Bourgain's embedding is based on randomly sampling sets in various densities, and defining the coordinates as distances to these sets. Our first observation (see Lemma 2) is sampling points only from a subset $K \subseteq X$ suffices to obtain an embedding which is nonexpansive for all pairs, and has bounded contraction for pairs in $K \times X$. Furthermore, the contraction depends only on $|K|$, rather than on $|X|$.

We then use a similar strategy as in previous sections, and partition X into roughly $\log \log n$ subsets $S_0, S_1, \ldots, S_{\log \log n}$, where S_i is of size $\approx 2^{2^i}$. The doubly exponential size arises because for any $u, v \in S_i$, the logarithm of the ranking of u and of v differs by at most a factor of 2. For each i, we create the embedding f_i that will "handle" pairs in $S_i \times X$, and concatenate all these functions $f = \bigoplus_{i=0}^{\log \log n} \alpha_i \cdot f_i$. Without the α_i factor, every pair will suffer a $(\log \log n)^{1/p}$ term in the distortion due to expansion. We introduce these factors into the embedding, where α_i is such that $\sum_{i=0}^\infty \alpha_i^p \leq 1$. In such a way, the function f is nonexpansive, but we pay a small factor of $1/\alpha_i$ in the distortion for pairs in $S_i \times X$.

LEMMA 2. *Let (X,d) be a metric space of size $|X|=n$, $K\subseteq X$ a subset of size $|K|=k$, and a parameter $p\in[1,\infty]$. Then there is a nonexpansive embedding of X into $\ell_p^{O(\log^2 k)}$ such that the contraction of any pair in $K\times X$ is at most $O(\log k)$.*

Proof. Let $m=O(\log^2 k)$, and $f:K\to\ell_p^m$ be a nonexpansive embedding with contraction $\delta=O(\log k)$ on the pairs of $K\times K$, which exists due to [Bou85, LLR95]. Let $\hat{f}$ be a nonexpansive extension to all of X as above. Let $h:X\to\mathbb{R}$ be defined by $h(x)=d(x,K)$. The embedding $F:X\to\ell_p^m$ is defined by the concatenation of these maps $F=\hat{f}\oplus h$. Since both of the maps $\hat{f},h$ are nonexpansive, it follows that for any $x,y\in X$,

$$\|F(x)-F(y)\|_p^p\le\|\hat{f}(x)-\hat{f}(y)\|_p^p+|h(x)-h(y)|^p\le 2\cdot d(x,y)^p\ ,$$

hence, F has expansion at most $2^{1/p}$ for all pairs. Let $t\in K$ and $x\in X$, and let $k_x\in K$ be such that $d(x,K)=d(x,k_x)$ (it could be that $k_x=x$). If it is the case that $d(x,t)\le 3\delta\cdot d(x,k_x)$, then by the single coordinate of h we get a sufficient contribution for this pair:

$$\|F(t)-F(x)\|_p\ge|h(t)-h(x)|=h(x)=d(x,k_x)\ge\frac{d(x,t)}{3\delta}\ .$$

The other case is that $d(x,t)>3\delta\cdot d(x,k_x)$, here we will get the contribution from $\hat{f}$. First observe that by the triangle inequality,

$$d(t,k_x)\ge d(t,x)-d(x,k_x)\ge d(t,x)(1-1/(3\delta))\ge 2d(t,x)/3\ . \tag{6}$$

By another application of the triangle inequality, using that $\hat{f}$ is nonexpansive, and that f has contraction δ on K, we get the required bound on the contraction:

$$\begin{aligned}\|F(t)-F(x)\|_p&\ge\|\hat{f}(t)-\hat{f}(x)\|_p\\&\ge\|\hat{f}(t)-\hat{f}(k_x)\|_p-\|\hat{f}(k_x)-\hat{f}(x)\|_p\\&\ge\|f(t)-f(k_x)\|_p-d(x,k_x)\\&\ge\frac{d(t,k_x)}{\delta}-\frac{d(t,x)}{3\delta}\\&\overset{(6)}{\ge}\frac{2d(t,x)}{3\delta}-\frac{d(t,x)}{3\delta}\\&=\frac{d(t,x)}{3\delta}\ .\end{aligned}$$

In particular, the function $2^{-\frac{1}{p}}\cdot F$ is nonexpansive for all pairs, and has contraction at most $2^{\frac{1}{p}}\cdot 3\cdot\delta=O(\log k)$ for pairs in $K\times X$. ∎

We are now ready to prove Theorem 14.

Proof of Theorem 14. Let $S_0=\{x_1,x_2\}$, and for $1\le i\le\lceil\log\log n\rceil$ let $S_i=\{x_j\ :\ 2^{2^{i-1}}<j\le 2^{2^i}\}$. For every i, let $f_i:X\to\ell_p$ be the embedding of Lemma 2 with $K=S_i$, and let $\alpha_i=c\cdot(i+1)^{-(1+\epsilon)/p}$ for sufficiently small constant c, so that $\sum_{i=0}^{\infty}\alpha_i^p\le 1$. Finally, define the embedding $f:X\to\ell_p$ by

$$f=\bigoplus_{i=0}^{\lceil\log\log n\rceil}\alpha_i\cdot f_i\ .$$

To see that f is indeed nonexpansive, recalling that each f_i is nonexpansive, we obtain that for any $u, v \in X$

$$\|f(u) - f(v)\|_p^p \leq \sum_{i=0}^{\lceil \log\log n \rceil} \alpha_i^p \cdot \|f_i(u) - f_i(v)\|_p^p \leq d(u,v)^p \sum_{i=0}^{\infty} \alpha_i^p \leq d(u,v)^p \ .$$

For the contraction, let $v = x_j$ for some $j > 1$, and take any $u \in X$. Let i be the index such that $v \in S_i$, and note that $2^{i-1} < \log j$. By Lemma 2, the embedding f_i has contraction at most $O(\log |S_i|) = O(2^i) = O(\log j)$ for the pair u, v. Observe that $\alpha_i^p = c^p \cdot (i+1)^{-(1+\epsilon)} = \Omega\left((2 + \log\log j)^{-(1+\epsilon)}\right)$, thus

$$\|f(u) - f(v)\|_p^p \geq \alpha_i^p \cdot \|f(u) - f(v)\|_p^p \geq \Omega\left(\frac{d(u,v)^p}{(\log j)^p \cdot (2 + \log\log j)^{-(1+\epsilon)}}\right) \ .$$

It is not hard to verify that x_1 has constant contraction with any u, so the prioritized distortion is $O\left(\log j \cdot (\log\log j)^{-(1+\epsilon)/p}\right)$. Finally, since the dimension of f_i is $O(\log^2 |S_i|) = O(2^{2i})$, the embedding f maps X into $\sum_{i=0}^{\lceil \log\log n \rceil} O(2^{2i}) = O(\log^2 n)$ dimensions. For $1 \leq p \leq 2$, one may embed first into ℓ_2, use [JL84] to reduce the dimension to $O(\log n)$, and then apply an embedding to $\ell_p^{O(\log n)}$, while paying a constant factor in the distortion [FLM77]. The prioritized distortion will thus be at most $O(\log j \cdot (\log\log j)^{(1+\epsilon)/2})$. ∎

8.2. Embedding with prioritized dimension. The main result of this section is an embedding with prioritized distortion *and dimension*. This means that a high ranking point will have low distortion (with any other point) and, additionally, its image will consist of few nonzero coordinates, followed by zeros in the rest.

THEOREM 15. *For any $p \in [1, \infty]$, $\epsilon > 0$, and any metric space (X, d) on n points, there exists an embedding of X into $\ell_p^{O(\log^2 n)}$ with priority distortion $O\left(\log^{4+\epsilon} j\right)$ and prioritized dimension $O(\log^4 j)$.*

Proof overview. The basic framework of this embedding appears at a first glance to be similar to section 8.1, which is applying a variation of Bourgain's embedding, while sampling only from certain subsets S_i of the points. However, the crux here is that we need to ensure that high priority points will be mapped to the zero vector in the embeddings that handle the lower ranked points.

Recall that the coordinates of the embedding are given by distances to sets. The idea is the following: while creating the embedding for the points in S_i, we insert all the points with higher ranking (those in $S_0 \cup \cdots \cup S_{i-1}$) into every one of the randomly sampled sets. This will certify that the high ranked points are mapped to zero in every one of these coordinates. However, the analysis of the distortion no longer holds, as the sets are not randomly chosen. Fix some point $u \in S_i$ and $v \in X$. The crucial observation is that if none of the higher ranked points lie in certain neighborhoods around u and v (the size of these neighborhoods depends on $d(u,v)$), then we can still use the randomness of the selected sets to obtain some bound (albeit not as good as the standard embedding achieves). While if there exists a high ranked point nearby, say $z \in S_{i'}$ for some $i' < i$, then we argue that u, v should already have sufficient contribution from the embedding designed for $S_{i'}$. The formal derivation of this idea is captured in Lemma 3.

The calculation shows that the distortion guarantee for u, v deteriorates by a logarithmic factor for each i, that is, it is the product of the distortion bound for points in S_{i-1} multiplied by $O(\log|S_i|)$. This implies that the optimal size of S_i is *triple exponential* in i, which yields the best balance between the price paid due to the size of S_i and the product of the logarithms of $|S_0|,\ldots,|S_{i-1}|$.

LEMMA 3. *Let $p\in[1,\infty]$ and $D\ge 1$. Given a metric space (X,d), two disjoint subsets $A,K\subseteq X$, where $|K|=k\ge 2$, and a nonexpansive embedding $g:X\to\ell_p$ with contraction at most D for all pairs in $A\times X$, then there is a nonexpansive embedding $f:X\to\ell_p^{O(\log^2 k)}$ such that the following properties hold:*

1. *For all $x\in A$, $f(x)=\vec{0}$.*
2. *For all $(x,y)\in K\times X$, $\|f(x)-f(y)\|_p\ge\frac{d(x,y)}{1000D\cdot\log k}$ or $\|g(x)-g(y)\|_p\ge\frac{d(x,y)}{2D}$.*

We postpone the proof of Lemma 3 to section 8.2.1, and prove Theorem 15 using the lemma.

Proof of Theorem 15. Let $I=\lceil\log\log\log n\rceil$. Let $S_0=\{x_1,x_2,x_3,x_4\}$, and for $1\le i\le I$ let

$$S_i=\left\{x_j\ :\ 2^{2^{2^{i-1}}}<j\le 2^{2^{2^i}}\right\}.$$

Also define $S_{<i}=\bigcup_{0\le k<i}S_k$.

The desired embedding $F:X\to\ell_p$ will be created by iteratively applying Lemma 3, each time using its output function f as part of the input for the next iteration. Formally, for each $0\le i\le I$ apply Lemma 3 with parameters $A=S_{<i}$, $K=S_i$, $g=F^{(i-1)}$, and $D=2^{2^i+5i^2}$, to obtain a map $f_i:X\to\ell_p$. The map $F^{(i)}:X\to\ell_p$ is defined as follows: $F^{(-1)}\equiv 0$ and $F^{(i)}=\bigoplus_{k=0}^{i}\alpha_k\cdot f_k$, where (α_k) is a sequence that ensures $F^{(i)}$ is nonexpansive for all i. For concreteness, take $\alpha_k=(\frac{6}{\pi^2(k+1)^2})^{1/p}$. The final embedding is defined by $F=F^{(I)}$.

Fix any pair $x,y\in X$. As f_i is nonexpansive by Lemma 3, we obtain that F is nonexpansive as well:

$$\|F(x)-F(y)\|_p^p=\sum_{i=0}^{I}\alpha_i^p\cdot\|f_i(x)-f_i(y)\|_p^p\le\sum_{i=0}^{\infty}\frac{6}{\pi^2(i+1)^2}\cdot d(x,y)^p=d(x,y)^p\ .$$

Next, we must show that for each $0\le i\le I$, the embedding $F^{(i-1)}$ has contraction at most $2^{2^i+5i^2}$ for pairs in $S_{<i}\times X$ to comply with the requirement of Lemma 3. We prove this by induction on i, the base case for $i=0$ holds trivially as $F^{(-1)}$ has no requirement on its contraction (since $S_{<0}=\emptyset$). Assume (for i) that $F^{(i-1)}$ has contraction at most $2^{2^i+5i^2}$ on pairs in $S_{<i}\times X$. For $i+1$, let $x\in S_{<i+1}$ and $y\in X$. Recall that $F^{(i)}$ is generated by applying Lemma 3 with $A=S_{<i}$, $K=S_i$, $g=F^{(i-1)}$, and $D=2^{2^i+5i^2}$. Then the lemma returns f_i, and finally $F^{(i)}=g\oplus(\alpha_i\cdot f_i)$.

We may assume that $x\in S_i$, otherwise $g=F^{(i-1)}$ has the required contraction on x,y by the induction hypothesis. Apply condition (2) of the lemma: if it is the case that $\|g(x)-g(y)\|_p\ge d(x,y)/(2D)$, then clearly $2D<2^{2^{i+1}+5(i+1)^2}$. The other case is that $\|f_i(x)-f_i(y)\|_p\ge\frac{d(x,y)}{1000D\cdot\log|S_i|}$. Since $\log|S_i|\le 2^{2^i}$ and $1/\alpha_i\le 2(i+1)^2$, the contraction of $F^{(i)}$ is at most the contraction of $\alpha_i\cdot f_i$, which is bounded by

$$\frac{1000D\cdot\log|S_i|}{\alpha_i}\le 1000\cdot 2^{2^i+5i^2}\cdot 2^{2^i}\cdot 2(i+1)^2<2^{2\cdot 2^i+5i^2+2\log(i+1)+11}<2^{2^{i+1}+5(i+1)^2}\ .$$

Observe that if $x=x_j\in S_i$ for some $j>1$, then $2^{2^{i-1}}<\log j$, and thus the distortion of F for any pair containing x is at most $2^{2^{i+1}+5(i+1)^2}=O(\log^4 j)\cdot$

$2^{O((2+\log\log\log j)^2)} = O(\log^{4+\epsilon} j)$. Additionally, note that as the distortion of $F^{(I-1)}$ is at most $D = 2^{2^I+5I^2}$, the same argument suggests that the maximal distortion of $F = F^{(I)}$ for any pair is at most

$$\frac{1000D \cdot \log n}{\alpha_I} \leq 1000 \cdot 2^{2^I+5I^2} \cdot \log n \cdot 2(I+1)^2 = O(\log^{3+\epsilon} n) \ .$$

Finally, let us bound the number of nonzero coordinates of the points. Recall that f_i maps X into $O(\log^2 |S_i|) \leq O(2^{2^{i+1}})$ dimensions. Fix some $x = x_j$ for $j > 1$, and let i be such that $x_j \in S_i$. Note that $2^{2^{i-1}} < \log j$, so that $2^{2^{i+1}} < \log^4 j$. By Lemma 3, for every $i' > i$, $f_{i'}(x_j) = \vec{0}$, and the number of coordinates used by $F^{(i)}$ is at most

$$\sum_{k=0}^{i} O(2^{2^{k+1}}) = O(2^{2^{i+1}}) = O(\log^4 j) \ .$$

Since the dimension of f_I is at most $O(\log^2 n)$, we get that the total number of coordinates used by F is only

$$\sum_{k=0}^{I-1} O(2^{2^{k+1}}) + O(\log^2 n) \leq O(2^{2^{1+\log\log\log n}}) + O(\log^2 n) = O(\log^2 n) \ .$$

□

8.2.1. Proof of Lemma 3. The basic approach to the proof is similar to Lemma 2, which is sampling subsets of K, according to various densities. The main difference is that we insert all the points of A into each sampled set, to ensure $f(x) = \vec{0}$ for all $x \in A$. The standard analysis of Bourgain for a pair x, y, considers certain neighborhoods defined according to the density of points around x, y. We show that the analysis still works as long as no point of A is present in those neighborhoods. Thus we can obtain a contribution which is proportional to the distance of x, y to A (or to $d(x, y)$ if that distance is large). This motivates the following definition and lemma.

DEFINITION 1. *The γ-distance between x and y with respect to A is defined to be*

$$\gamma_A(x, y) = \min\left\{\frac{d(x, y)}{2}, d(x, A), d(y, A)\right\} \ .$$

LEMMA 4. *Let $c = 24$. There exists a nonexpansive embedding $\varphi : X \to \ell_p^{O(\log^2 k)}$, such that for all $z \in A$, $\varphi(z) = \vec{0}$, and for all $x, y \in K$,*

$$\|\varphi(x) - \varphi(y)\|_p \geq \frac{\gamma_A(x, y)}{c \log k} \ .$$

We defer the proof of Lemma 4, and proceed first with the proof of Lemma 3. Define $h : X \to \mathbb{R}$ for $x \in X$ as $h(x) = d(x, A \cup K)$. Our embedding f is

$$f = \frac{\varphi \oplus h}{2^{1/p}} \ .$$

Since both φ and h are nonexpansive and vanish on A, clearly f is nonexpansive as well, and $f(z) = \vec{0}$ for any $z \in A$. It remains to show property (2) of the lemma. Fix any $x \in K$ and $y \in X$, and consider the following three cases:

Case 1. $d(\{x,y\},A) \leq \frac{d(x,y)}{4D}$.

In this case we shall use the guarantees of the map g. Assume w.l.o.g. that $z \in A$ is such that $d(y,z) \leq \frac{d(x,y)}{4D}$. Then by the triangle inequality

$$d(x,z) \geq d(x,y) - d(y,z) \geq d(x,y) - \frac{d(x,y)}{4D} \geq \frac{3d(x,y)}{4} . \tag{7}$$

Now, using that g is nonexpansive, and has contraction at most D for any pair in $A \times X$, we obtain that

$$\begin{aligned} \|g(x)-g(y)\|_p &\geq \|g(x)-g(z)\|_p - \|g(z)-g(y)\|_p \\ &\geq \frac{d(x,z)}{D} - d(z,y) \\ &\overset{(7)}{\geq} \frac{3d(x,y)}{4D} - \frac{d(x,y)}{4D} \\ &= \frac{d(x,y)}{2D} , \end{aligned}$$

which satisfies property (2).

Case 2. $d(\{x,y\},A) > \frac{d(x,y)}{4D}$ and $d(y,K) \geq \frac{d(x,y)}{20cD\cdot\log k}$ (where $c = 24$ is the constant of Lemma 4).

Here we shall use the map h for the contribution. Since $d(y,A) \geq d(x,y)/(4D)$, we have that $h(y) = d(y, A \cup K) \geq \frac{d(x,y)}{20cD\cdot\log k}$ and of course $h(x) = 0$, so that

$$\|f(x)-f(y)\|_p \geq \frac{|h(x)-h(y)|}{2} \geq \frac{d(x,y)}{40cD\cdot\log k} ,$$

as required.

Case 3. $d(\{x,y\},A) > \frac{d(x,y)}{4D}$ and $d(y,K) < \frac{d(x,y)}{20cD\cdot\log k}$.

In this case, the function φ will yield the required contribution, by employing a similar strategy to Lemma 2. Let $k_y \in K$ be such that $d(y,k_y) = d(y,K)$. Note that $d(k_y,A) \geq d(y,A) - d(y,k_y) \geq \frac{d(x,y)}{4D} - \frac{d(x,y)}{20cD\cdot\log k} \geq \frac{d(x,y)}{5D}$, and it follows that

$$\gamma_A(x,k_y) \geq \frac{d(x,y)}{5D} . \tag{8}$$

By Lemma 4, since f is nonexpansive, and using another application of the triangle inequality, we conclude that

$$\begin{aligned} \|f(x)-f(y)\|_p &\geq \|f(x)-f(k_y)\|_p - \|f(y)-f(k_y)\|_p \\ &\geq \frac{\|\varphi(x)-\varphi(k_y)\|_p}{2} - d(y,k_y) \\ &\geq \frac{\gamma_A(x,k_y)}{2c\log k} - \frac{d(x,y)}{20cD\cdot\log k} \\ &\overset{(8)}{\geq} \frac{d(x,y)}{10cD\cdot\log k} - \frac{d(x,y)}{20cD\cdot\log k} \\ &= \frac{d(x,y)}{20cD\cdot\log k} . \end{aligned}$$

This concludes the proof of Lemma 3. It remains to validate Lemma 4, which is similar in spirit to the methods of [Bou85, LLR95]; we give full details for completeness.

Proof of Lemma 4. Let $I = \lceil \log k \rceil$ and $J = C \cdot \log k$ for a constant C that will be determined later. For each $i \in [I]$ and $j \in [J]$ sample a set Q'_{ij} by including each $x \in K$ independently with probability 2^{-i}, and let $Q_{ij} = Q'_{ij} \cup A$. Define maps $\varphi_{ij} : X \to \mathbb{R}$ by letting for each $u \in X$, $\varphi_{ij}(u) = d(u, Q_{ij})$, and $\varphi : X \to \ell_p^{I \cdot J}$ by

$$\varphi(u) = \frac{1}{(I \cdot J)^{1/p}} \bigoplus_{i \in [I]} \bigoplus_{j \in [J]} \varphi_{ij}(u) \ .$$

Since each φ_{ij} is nonexpansive, φ is nonexpansive as well, and in what follows we bound its contraction.

Define for $u \in K$ and $r \geq 0$ the ball restricted to K, $B_K(u, r) = B(u, r) \cap K$, and recall that by B° we mean the open ball. Fix a pair $u, v \in K$, and for each $0 \leq i \leq I$, let r'_i be the minimal such that both $|B_K(u, r)| \geq 2^i$ and $|B_K(v, r)| \geq 2^i$. Define $r_i = \min\{r'_i, \gamma_A(u, v)\}$ and let $\Delta_i = r_i - r_{i-1}$. Observe that $r_0 = 0$ and $r_I = \gamma_A(u, v)$, so that

$$\sum_{i \in [I]} \Delta_i = \gamma_A(u, v) \ . \tag{9}$$

We first claim that for each $i \in [I]$ and $j \in [J]$,

$$\Pr[|\varphi_{ij}(u) - \varphi_{ij}(v)| \geq \Delta_i] \geq 1/12 \ . \tag{10}$$

If $\Delta_i = 0$ then there is nothing to prove. Assume then that $r_{i-1} < r_i$, and note that either $|B_K^\circ(u, r_i)| \leq 2^i$ or $|B_K^\circ(v, r_i)| \leq 2^i$ (otherwise it contradicts the minimality of r_i). W.l.o.g. we have that $|B_K^\circ(u, r_i)| \leq 2^i$. Furthermore, note that the sets $B_K^\circ(u, r_i)$, $B_K(v, r_{i-1})$, and A are pairwise disjoint. Let $\mathcal{E}$ be the event that $\{Q_{ij} \cap B_K^\circ(u, r_i) = \emptyset\}$ and $\mathcal{F}$ be the event that $\{Q_{ij} \cap B_K(v, r_{i-1}) \neq \emptyset\}$. Observe that if both events hold then $d(u, Q_{ij}) \geq r_i$ and $d(v, Q_{ij}) \leq r_{i-1}$, so that

$$|\varphi_{ij}(u) - \varphi_{ij}(v)| \geq r_i - r_{i-1} = \Delta_i \ .$$

Since both balls are disjoint from A, we have that

$$\Pr[\mathcal{E}] = \prod_{x \in B_K^\circ(u, r_i)} \Pr\left[x \notin Q'_{ij}\right] = \left(1 - 2^{-i}\right)^{|B_K^\circ(u, r_i)|} \geq \left(1 - 2^{-i}\right)^{2^i} \geq \frac{1}{4} \ .$$

And similarly,

$$\begin{aligned}\Pr[\mathcal{F}] &= 1 - \prod_{x \in B_K(v, r_{i-1})} \Pr\left[x \notin Q'_{ij}\right] = 1 - \left(1 - 2^{-i}\right)^{|B_K(v, r_{i-1})|} \\ &\geq 1 - \left(1 - 2^{-i}\right)^{2^{i-1}} \geq 1 - e^{-\frac{1}{2}} \geq \frac{1}{3} \ .\end{aligned}$$

Since the events $\mathcal{E}$ and $\mathcal{F}$ are independent, this concludes the proof of (10). Let X_{ij} be an indicator random variable for the event that $|\varphi_{ij}(u) - \varphi_{ij}(v)| \geq \Delta_i$, and $X_i =$

$\sum_{j=1}^{J} X_{ij}$. Using the independence for different values of j, and that $\mathbb{E}[X_i] \geq J/12$, a Chernoff bound yields that for any i

$$\Pr[X_i < J/24] \leq e^{-J/100} \leq 1/k^3 ,$$

when C is sufficiently large. Note that if indeed $X_i \geq J/24$ for all $1 \leq i \leq I$, then

$$\begin{aligned}\|\varphi(u) - \varphi(v)\|_p^p &= \frac{1}{I \cdot J} \sum_{i=1}^{I} \sum_{j=1}^{J} |\varphi_{ij}(u) - \varphi_{ij}(v)|^p \\ &\geq \frac{1}{24I} \sum_{i=1}^{I} \Delta_i^p \\ &\geq \frac{I^{1-p}}{24I} \Big(\sum_{i=1}^{I} \Delta_i \Big)^p \\ &\overset{(9)}{\geq} \frac{\gamma_A(u,v)^p}{24I^p} ,\end{aligned}$$

where the second inequality uses Hölder's inequality. Applying a union bound over the $\binom{k}{2}$ possible pairs in $\binom{K}{2}$, and the $I = \lceil \log k \rceil$ possible values of i, there is at least a constant probability that for every pair $\|\varphi(u) - \varphi(v)\|_p \geq \frac{\gamma_A(u,v)}{24^{1/p} \cdot \log k}$. ∎

REFERENCES

[ABC+05] I. Abraham, Y. Bartal, H. T.-H. Chan, K. Dhamdhere, A.Gupta, J. M. Kleinberg, O. Neiman, and A.s Slivkins, *Metric embeddings with relaxed guarantees*, in Proceedings of the 46th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2005), Pittsburgh, PA, IEEE Computer Society, Los Alamitos, CA, 2005, pp. 83–100.

[ABN07] I. Abraham, Y. Bartal, and O. Neiman, *Embedding metrics into ultrametrics and graphs into spanning trees with constant average distortion*, in Proceedings of the 18th Annual ACM-SIAM Symposium on Discrete Algorithms, SODA '07, 2007, SIAM, Philadelphia, pp. 502–511.

[ABN11] I. Abraham, Y. Bartal, and O. Neiman, *Advances in metric embedding theory*, Adv. Math., 228 (2011), pp. 3026–3126.

[AC14] I. Abraham and S. Chechik, *Distance labels with optimal local stretch*, in Proceedings of the 41st International Colloquium on Automata, Languages, and Programming, ICALP 2014, Copenhagen, Denmark, 2014, Part I, Springer, Heidelberg, 2014, pp. 52–63.

[AG06] I. Abraham and C. Gavoille, *Object location using path separators*, in Proceedings of the Twenty-Fifth Annual ACM Symposium on Principles of Distributed Computing, PODC 2006, Denver, CO, 2006, ACM, New York, 2006, pp. 188–197.

[AP92] B. Awerbuch and D. Peleg, *Routing with polynomial communication-space tradeoff*, SIAM J. Discrete Math., 5 (1992), pp. 151–162.

[Bar96] Y. Bartal, *Probabilistic approximation of metric spaces and its algorithmic applications*, in Proceedings of the 37th Annual Symposium on Foundations of Computer Science, FOCS '96, IEEE Computer Society, Los Alamitos, CA, 1996, pp. 184–193.

[Bar98] Y. Bartal, *On approximating arbitrary metrices by tree metrics*, in Proceedings of the 30th Annual ACM Symposium on Theory of Computing, STOC '98, NY, ACM, New York, 1998, pp. 161–168.

[BBMN11] N. Bansal, N. Buchbinder, A. Madry, and J. Naor, *A polylogarithmic-competitive algorithm for the k-server problem*, in Proceedings of the 52th Annual IEEE Symposium on Foundations of Computer Science, FOCS '08, IEEE, Piscataway, NJ, 2011, pp 267 –276.

[BFN16] Y. Bartal, A. Filtser, and O.Neiman, *On notions of distortion and an almost minimum spanning tree with constant average distortion*, in Proceedings of the

Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA '16, SIAM, Philadelphia, 2016, pp. 873–882.

[Bou85] J. Bourgain, *On Lipschitz embedding of finite metric spaces in Hilbert space*, Israel J. Math., 52 (1985), pp. 46–52.

[Bou86] J. Bourgain, *The metrical interpretation of superreflexivity in Banach spaces*, Israel J. Math., 56 (1986), pp. 222–230.

[CDG06] H. T.-H. Chan, M. Dinitz, and A. Gupta, *Spanners with slack*, in Proceedings of the 14th Annual European Symposium, Algorithms - ESA 2006, Zurich, Switzerland, Springer, Berlin, 2006, pp. 196–207.

[Che14] S. Chechik, *Approximate distance oracles with constant query time*, in Proceedings of the 46th Annual ACM Symposium on Theory of Computing, STOC '14, NY, ACM, New York, 2014, pp. 654–663.

[Che15] S. Chechik, *Approximate distance oracles with improved bounds*, in Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing, STOC 2015, Portland, OR, 2015, ACM, New York, 2015, pp. 1–10.

[EFN15] M. Elkin, A. Filtser, and O. Neiman, *Prioritized metric structures and embedding*, in Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing, STOC 2015, Portland, OR, ACM, New York, 2015, pp. 489–498.

[FLM77] T. Figiel, J. Lindenstrauss, and V. D. Milman, *The dimension of almost spherical sections of convex bodies*, Acta Math., 139 (1977), pp. 53–94.

[FRT04] J. Fakcharoenphol, S. Rao, and K. Talwar, *A tight bound on approximating arbitrary metrics by tree metrics*, J. Comput. System Sci., 69 (2004), pp. 485–497.

[GPPR01] C. Gavoille, D. Peleg, S. Perennes, and R. Raz, *Distance labeling in graphs*, in Proceedings of the Twelfth Annual ACM-SIAM Symposium on Discrete Algorithms, 2001, SIAM, Philadelphia, 2001, pp. 210–219.

[Gup01] A. Gupta, *Steiner points in tree metrics don't (really) help*, in Proceedings of the Twelfth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA '01, SIAM, Philadelphia, 2001, pp. 220–227.

[JL84] W. B. Johnson and J. Lindenstrauss, *Extensions of Lipschitz mappings into a Hilbert space*, in Conference in Modern Analysis and Probability (New Haven, CT, 1982), AMS, Providence, RI, 1984, pp. 189–206.

[KKM+12] M. Khan, F. Kuhn, D. Malkhi, G. Pandurangan, and K. Talwar, *Efficient distributed approximation algorithms via probabilistic tree embeddings*, Distrib. Comput., 25 (2012), pp. 189–205.

[Kle02] P. N. Klein, *Preprocessing an undirected planar network to enable fast approximate distance queries*, in Proceedings of the Thirteenth Annual ACM-SIAM Symposium on Discrete Algorithms, 2002, San Francisco, CA, SIAM, Philadelphia, 2002, pp. 820–827.

[KSW09] J. Kleinberg, A. Slivkins, and T. Wexler, *Triangulation and embedding using small sets of beacons*, J. ACM, 56 (2009), 32.

[LLR95] N. Linial, E. London, and Y. Rabinovich, *The geometry of graphs and some of its algorithmic applications*, Combinatorica, 15 (1995), pp. 215–245.

[LT79] R. J. Lipton and R. E. Tarjan, *A separator theorem for planar graphs*, SIAM J. Appl. Math., 36 (1979), pp. 177–189.

[MN06] M. Mendel and A. Naor, *Ramsey partitions and proximity data structures*, in Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science, FOCS '06, IEEE Computer Society, Los Alamitos, CA, 2006, pp. 109–118.

[Pel99] D. Peleg, *Proximity-preserving labeling schemes and their applications*, in Proceedings of the Graph-Theoretic Concepts in Computer Science, 25th International Workshop, WG '99, Ascona, Switzerland, 1999, Springer, Berlin, 1999, pp. 30–41.

[RR98] Y. Rabinovich and R. Raz, *Lower bounds on the distortion of embedding finite metric spaces in graphs*, Discrete Comput. Geom., 19 (1998), pp. 79–94.

[RTZ05] L. Roditty, M.Thorup, and U. Zwick, *Deterministic constructions of approximate distance oracles and spanners*, in Proceedings of the 32nd International Conference on Automata, Languages and Programming, ICALP'05, Springer, Berlin, 2005, pp. 261–272.

[SS09] G. Schechtman and A. Shraibman, *Lower bounds for local versions of dimension reductions*, Discrete Comput. Geom., 41, (2009), pp. 273–283.

[ST04] Y. Shavitt and T. Tankel, *Big-bang simulation for embedding network distances in Euclidean space*, IEEE/ACM Trans. Netw., 12 (2004), pp. 993–1006.

[Tho01] M. Thorup, *Compact oracles for reachability and approximate distances in planar digraphs*, in Proceedings of the 42nd Annual Symposium on Foundations of Computer Science, FOCS 2001, Las Vegas, NV, IEEE Computer Society, Los Alamitos, CA, 2001, pp. 242–251.

[TZ01] M. Thorup and U. Zwick, *Compact routing schemes*, in Proceedings of the Thirteenth Annual ACM Symposium on Parallel Algorithms and Architectures, SPAA '01 New York, ACM, New York, 2001, pp. 1–10.

[TZ05] M. Thorup and U. Zwick, *Approximate distance oracles*, J. ACM, 52 (2005), pp. 1–24.

[Wul13] C. Wulff-Nilsen, *Approximate distance oracles with improved query time*, in Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2013, New Orleans, LA, 2013, SIAM, Philadelphia, 2013, pp. 539–549.

Part III

On Notions of Distortion and an Almost Minimum Spanning Tree with Constant Average Distortion

Journal of Computer and System Sciences 105 (2019) 116–129

Contents lists available at ScienceDirect

Journal of Computer and System Sciences

www.elsevier.com/locate/jcss

On notions of distortion and an almost minimum spanning tree with constant average distortion ☆

Yair Bartal [a,*,1], Arnold Filtser [b,*,2], Ofer Neiman [b,*,2]

[a] School of Engineering and Computer Science, Hebrew University, Israel
[b] Department of Computer Science, Ben-Gurion University of the Negev, Israel

A R T I C L E I N F O

Article history:
Received 5 February 2018
Received in revised form 3 February 2019
Accepted 25 April 2019
Available online 4 May 2019

Keywords:
Metric embedding
Prioritized distortion
Scaling distortion
Average distortion
Light spanner

A B S T R A C T

This paper makes two main contributions: a construction of a *near-minimum spanning tree* with *constant average distortion*, and a general equivalence theorem relating two refined notions of distortion: *scaling distortion* and *prioritized distortion*. *Scaling distortion* provides improved distortion for $1-\epsilon$ fractions of the pairs, for all ϵ simultaneously. A stronger version called *coarse* scaling distortion, has improved distortion guarantees for the furthest pairs. *Prioritized distortion* allows to prioritize the nodes whose associated distortions will be improved. We show that prioritized distortion is essentially equivalent to coarse scaling distortion via a general transformation. This equivalence is used to construct the near-minimum spanning tree with constant average distortion, and has many further implications to metric embeddings theory. Among other results, we obtain a strengthening of Bourgain's theorem on embedding arbitrary metrics into Euclidean space, possessing optimal prioritized distortion.

1. Introduction

One of the fundamental problems in graph theory is that of constructing a Minimum Spanning Tree (MST) of a given weighted graph $G=(V,E)$. This problem and its variants received much attention, and has found numerous applications. In many of these applications, one may desire not only minimizing the weight of the spanning tree, but also other desirable properties, at the price of losing a small factor in the weight of the tree compared to that of the MST. Define the *lightness* of T to be the total weight of T (the sum of its edge weights) divided by the weight of an MST. One well known example is that of a Shallow Light Tree (SLT) [27,6], which is a rooted spanning tree having near optimal $(1+\rho)$ lightness, while approximately preserving all distances from the root to the other vertices.

It is natural to ask that the spanning tree will preserve well all pairwise distances in the graph. However, it is easy to see that no spanning tree can maintain such a requirement. In particular, even in the case of the unweighted cycle graph on n vertices, for every spanning tree there is a pair of neighboring vertices whose distance increases by a factor of $n-1$. A natural relaxation of this demand is that the spanning tree approximates all pairwise distances *on average*. Formally, the distortion of the pair $u,v\in V$ in T is defined as $\frac{d_T(u,v)}{d_G(u,v)}$, and the *average distortion* is $\frac{1}{\binom{n}{2}}\sum_{\{u,v\}\in\binom{V}{2}}\frac{d_T(u,v)}{d_G(u,v)}$, where d_G

☆ Preliminary version of this paper was published in SODA'16 [13].

* Corresponding authors.
E-mail addresses: yair@cs.huji.ac.il (Y. Bartal), arnoldf@cs.bgu.ac.il (A. Filtser), neimano@cs.bgu.ac.il (O. Neiman).

1 Supported in part by a grant from the ISF (1817/17).
2 Supported in part by ISF grant No. (1817/17) and by BSF grant No. 2015813.

https://doi.org/10.1016/j.jcss.2019.04.006
0022-0000/© 2019 Elsevier Inc. All rights reserved.

(respectively d_T) is the shortest-path metric in G (resp. T).[3] There are graphs where the average distortion of the MST can be as large as $\Omega(n)$ (see for example the graph described in Lemma 13). In [4], it was shown that for every weighted graph, it is possible to find a spanning tree which has constant average distortion. However, the question of whether it is possible to obtain a tree with non-trivial bounds on both its lightness and the average distortion remained open.

In this paper we resolve this question in the affirmative. Specifically, we devise a spanning tree of optimal $(1+\rho)$ lightness that has $O(1/\rho)$ average distortion over all pairwise distances. We show that this result is tight by exhibiting a lower bound on the tradeoff between lightness and average distortion, that in order to get $1+\rho$ lightness the average distortion must be $\Omega(1/\rho)$ (this holds for $1/n \le \rho \le 1$, and even if the spanning subgraph is not necessarily a tree).

Our main result of a light spanning tree with constant average distortion may be of interest for network applications. It is extremely common in the area of distributed computing that an MST is used for communication between the network nodes. This allows easy centralization of computing processes and an efficient way of broadcasting through the network, allowing communication to all nodes at a minimum cost. Yet, as already mentioned above, when communication is required between specific pairs of nodes, the cost of routing through the MST may be extremely high, even when their real distance is small. However, in practice it is the average distortion, rather than the worst-case distortion, that is often used as a practical measure of quality, as has been a major motivation behind the initial work of [28,3,4]. As noted above, the MST still fails even in this relaxed measure. Our result overcomes this by promising small routing cost between nodes on average, while still possessing the low cost of broadcasting through the tree, thereby maintaining the standard advantages of the MST.

Our main result on a low average distortion embedding follows from analyzing the *scaling distortion* [28,3] of the embedding. This refined notion of distortion turns out to be closely related to another useful measure of *prioritized distortion* [22]. The second main contribution of this paper is providing an equivalence theorem stating the relation between these useful notions.

1.1. Scaling distortion vs. prioritized distortion: a general equivalence theorem

Scaling distortion, first introduced in [28],[4] requires that for *every* $0 < \epsilon < 1$, the distortion of all but an ϵ-fraction of the pairs is bounded by the appropriate function of ϵ. In [3] it was shown that one may obtain bounds on the average distortion, as well as on higher moments of the distortion function, from bounds on the scaling distortion. In [3] several scaling distortion results were shown including $O(\log(1/\epsilon))$ scaling distortion embedding into Euclidean space, and in [4] an $O(1/\sqrt{\epsilon})$ scaling embedding into trees, and spanning trees in particular.

Prioritized distortion, introduced recently in [22], requires that for every given ranking $v_1,\dots,v_n$ of the vertices of the graph, there is an embedding where the distortion of pairs including v_j is bounded as a function $\alpha(j):[n]\to\mathbb{R}_+$ of the rank j. Several prioritized distortion results were given in [22], including $\tilde{O}(\log j)$[5] prioritized distortion embedding into Euclidean space. (See Section 2 for formal definitions of embeddings and various notions of distortion.)

One of the main ingredients of our work is a *general reduction* relating the notions of prioritized distortion and scaling distortion. In fact, we show that prioritized distortion is essentially equivalent to a strong version of scaling distortion called *coarse* scaling distortion, in which for every point p and every $0<\epsilon<1$, the distances to the $1-\epsilon$ fraction of the farthest points from p are preserved with the desired distortion. We prove that any embedding with a prioritized distortion α has coarse scaling distortion bounded by $O(\alpha(8/\epsilon))$. This result could be of independent interest; in particular, it shows that the results of [22] have their scaling distortion counterparts (some of which were not known before). We further show a reduction in the opposite direction, informally, that given an embedding with coarse scaling distortion γ, there exists an embedding with prioritized distortion $\gamma(\mu(j))$, where μ is a function such that $\sum_i \mu(i)=1$ (e.g. $\mu(j)=\frac{6}{(\pi\cdot j)^2}$). We note that this reduction heavily relied on the property of coarse scaling distortion embeddings and does not apply to non-coarse scaling embeddings. Yet, most existing scaling embeddings are indeed coarse. This result implies that all existing *coarse* scaling distortion results have priority distortion counterparts, thus improving few of the results of [22]. In particular, by applying a theorem of [3] we obtain prioritized embedding of arbitrary metric spaces into l_p in dimension $O(\log n)$ and prioritized distortion $O(\log j)$, which exhibits a strengthening of Bourgain's theorem [15] (which asserts $O(\log n)$ worst-case distortion), and is best possible. It also implies better bounds for decomposable metrics (see [3]), such as planar and doubling metrics, where we obtain an optimal $O(\sqrt{\log j})$ prioritized distortion.

We also show an equivalence between embeddings with coarse partial distortion and terminal embeddings, which can be used to extend and improve previous results. See Section 3.3 for details.

In the context of our main construction of a light spanning tree, the first direction of the above equivalence theorem allows us to devise prioritized distortion embeddings and use these to obtain scaling distortion embeddings which possess the desired constant average distortion.

---

[3] Distortion is sometimes referred to as stretch.

[4] Originally coined gracefully degrading embedding.

[5] By $\tilde{O}(f(n))$ we mean $O(f(n)\cdot \mathrm{polylog}(f(n)))$.

1.2. Light spanning tree of constant average distortion

Our main spanning tree construction provides a spanning tree with $1+\rho$ lightness and scaling distortion $\tilde{O}(1/\sqrt{\epsilon})/\rho$, which is nearly tight as a function of ϵ [4]. This result implies that the average distortion is $O(1/\rho)$.

We also devise a probabilistic embedding: a distribution over spanning trees, each tree in the support of the distribution has $1+\rho$ lightness, and the expected scaling distortion is at most $\text{polylog}(1/\epsilon)/\rho$. Thus providing constant bounds on all fixed moments of the distortion (i.e., the l_q-distortion [3] for fixed q). (See the end of Section 5 for a formal definition of probabilistic embedding.)

Our main technical contribution, en route to this result, may be of its own interest: We devise a *spanner* (a subgraph of G) with $1+\rho$ lightness and low *prioritized distortion*. Here we show a light spanner construction with prioritized distortion at most $\tilde{O}(\log j)/\rho$. Using the equivalence theorem relating prioritized distortion and scaling distortion (discussed above), we obtain a spanner having scaling distortion $\tilde{O}(\log(1/\epsilon))/\rho$, and thus average distortion $O(1/\rho)$. Although we do not obtain a spanning tree here, this result has a few advantages, as we get constant bounds on all fixed moments of the distortion function (the ℓ_q-distortion). Moreover, the worst-case distortion is only logarithmic in n. We note that all of our results admit deterministic polynomial time algorithms.

Another technical contribution is a general, black-box reduction, that transform constructions of spanners with distortion t and lightness ℓ into spanners with distortion t/δ and lightness $1+\delta\ell$ (here $0<\delta<1$). This reduction can be applied in numerous settings, and also for many different special families of graphs. In particular, this reduction allows us to construct prioritized spanners with lightness arbitrarily close to 1.

Outline and techniques. Our proof has the following high level approach; Given a graph and a ranking of its vertices, we first find a low weight spanner with prioritized distortion $\tilde{O}(\log j)/\rho$. We then apply the general reduction from prioritized distortion to scaling distortion to find a spanner with scaling distortion $\tilde{O}(\log(1/\epsilon))/\rho$. Finally, we use the result of [4] to find a spanning tree of this spanner with scaling distortion $O(1/\sqrt{\epsilon})$. We then conclude that the scaling distortion of the composed embeddings[6] is roughly their product, which implies our main result of a spanning tree with lightness $1+\rho$ and scaling distortion $\tilde{O}(1/\sqrt{\epsilon})/\rho$.

Similarly, we can apply the probabilistic embedding of [4] to get a light counterpart, devising a distribution over spanning trees, each with lightness $1+\rho$, with (expected) scaling distortion $\text{polylog}(1/\epsilon)/\rho$.

The main technical part of the paper is finding a light prioritized spanner. In a recent result [19] (following [23,17]), it was shown that any graph on n vertices admits a spanner with (worst-case) distortion $O(\log n)$ and with constant lightness. However, these constructions have no bound on the more refined notions of distortion. To obtain a prioritized distortion, we use a technique similar in spirit to [22]: group the vertices into $\log\log n$ sets according to their priority, the set K_i will contain vertices with priority up to 2^{2^i}. We then build a low weight spanner for each of these sets. As prioritized distortion guarantees a bound for *every pair* containing a high ranking vertex, we must augment the spanner of K_i with shortest paths to all other vertices. Such a shortest path tree may have large weight, so we use an idea from [16] and apply an SLT rooted at K_i, which balances between the weight and the distortion from K_i.

The main issue with the construction described above is that the weight of the spanner in each phase can be proportional to that of the MST, but we have $\log\log n$ of those. Obtaining constant lightness, completely independent of n, requires a subtler argument. We use the fact that the weight of the light spanners in each phase comes "mostly" from the MST, and then some additional weight. We ensure that all the spanners will have *the same* MST. Then we select the parameters carefully, so that the additional weights will be small enough to form converging sequences, without affecting the distortion by too much.

1.3. Related work

Partial and scaling embeddings[7] have been studied in several papers [28,1,3,16,4,5]. Some of the notable results are embedding arbitrary metrics into a distribution over trees [1] or into Euclidean space [3] with tight $O(\log(1/\epsilon))$ scaling distortion. The notion ℓ_q-distortion was introduced in [3], they show that their scaling distortion results imply constant average distortion and $O(q)$ bound on the ℓ_q-distortion. This notion has been further studied in several papers, including [4, 5,16], and most recently applied in the context of dimensionality reduction [14]. In [4], an embedding into a single spanning tree with tight $O(1/\sqrt{\epsilon})$ scaling distortion was shown, which implies, in particular, constant average distortion, but there is no guarantee on the weight of the tree. It follows from [1] that this bound is tight even when embedding into arbitrary (non-spanning) trees.

Prioritized distortion embeddings were studied in [22], for instance they give an embedding of arbitrary metrics into a distribution over trees with expected prioritized distortion $O(\log j)$, and into Euclidean space with prioritized distortion $\tilde{O}(\log j)$.

[6] Given two embeddings $f: X \to Y$, $g: Y \to Z$, the composition $g \circ f$ is defined from X to Z, sending every point $x \in X$ to $g(f(x)) \in Z$.

[7] A partial embedding (introduced by [28] under the name *embedding with slack*) requires that for a *fixed* $0<\epsilon<1$, the distortion of all but an ϵ-fraction of the pairs is bounded by the appropriate function of ϵ.

Probabilistic embedding into trees [10–12,24] and spanning trees [8,20,2,9] has been intensively studied, and found numerous applications to approximation and online algorithms, and to fast linear system solvers. While our distortion guarantee does not match the best known worst-case bounds, which are $O(\log n)$ for arbitrary trees [12,24] and $\tilde{O}(\log n)$ for spanning trees [2,9], we give the first probabilistic embeddings into spanning trees with polylogarithmic scaling distortion in which all the spanning trees in the support of the distribution are light.

The paper [16] considers partial and scaling embedding into spanners, and show a general transformation from worst-case distortion to partial and scaling distortion. In particular, they show a spanner with $O(n)$ edges and $O(\log(1/\epsilon))$ scaling distortion. For a fixed $\epsilon > 0$, they also obtain a spanner with $O(n)$ edges, $O(\log(1/\epsilon))$ partial distortion and lightness $O(\log(1/\epsilon))$.[8] Note that these results fall short of achieving both constant average distortion and constant lightness.

In a subsequent work, [25] used our general reduction for light spanners (Theorem 8), to show that the $O(\log n/\rho)$-greedy spanner has lightness $1+\rho$.

2. Preliminaries

All the graphs $G = (V, E, w)$ we consider are undirected and weighted with nonnegative weights. Throughout the paper we denote by n the number of vertices and by m the number of edges in the given graph. We shall assume w.l.o.g. that all edge weights are different. If it is not the case, then one can break ties in an arbitrary (but consistent) way. Note that under this assumption, the MST T of G is unique. The weight of a graph G is $w(G) = \sum_{e\in E} w(e)$. Let d_G be the shortest path metric on G. For a subset $K \subseteq V$ and $v \in V$ let $d_G(v, K) = \min_{u\in K}\{d_G(u, v)\}$. For $r \geq 0$ let $B_G(v,r) = \{u \in V : d_G(u, v) \leq r\}$ (we often omit the subscript when clear from context).

For a graph $G = (V, E)$ on n vertices, a subgraph $H = (V, E')$ where $E' \subseteq E$ (with the induced weights) is called a *spanner* of G. We say that a pair $u, v \in V$ has *distortion* at most t if

$$d_H(v,u) \leq t \cdot d_G(v,u)\ ,$$

(note that always $d_G(v,u) \leq d_H(v,u)$). If every pair $u, v \in V$ has distortion at most t, we say that the spanner H has distortion t. Let T be the (unique) MST of G, the *lightness* of H is the ratio between the weight of H and the weight of the MST, that is $\Psi(H) = \frac{w(H)}{w(T)}$. We sometimes abuse notation and identify a spanner or a spanning tree with its set of edges.

A metric space (X, d_X) is defined over a set of points X and a nonnegative distance function d_X, with positive values on distinct points, and obeying the triangle inequality. Every weighted graph G can be viewed as a metric space (V, d_G). For two metric spaces (X, d_X), (Y, d_Y) and a non-contractive embedding $f : X \to Y$,[9] the distortion of a pair $x, y \in X$ under f is defined as $\frac{d_Y(f(x),f(y))}{d_X(x,y)}$.

When considering a graph G and its subgraph H, we may view the metric of G as being embedded into H via the identity map, in which case the last definition of distortion given above coincides with the those given earlier. Hence, the following definitions may be interpreted in the graph case in the obvious way.

Prioritized distortion. Let $(X, d_X), (Y, d_Y)$ be metric spaces. Let $\pi = v_1, \ldots, v_n$ be a priority ranking (an ordering) of the points (vertices) of X, and let $\alpha : \mathbb{N} \to \mathbb{R}_+$ be some monotone non-decreasing function. We say that a non-contractive embedding $f : X \to Y$ has *prioritized distortion* α (w.r.t. π), if for all $1 \leq j < i \leq n$, the pair v_j, v_i has distortion (under f) at most $\alpha(j)$.

Scaling distortion. Let $(X, d_X), (Y, d_Y)$ be metric spaces, with $|X| = n$. For $v \in X$ and $\epsilon \in (0,1)$ let $R(v,\epsilon) = \min\{r : |B(v,r)| \geq \epsilon n\}$. A point $u \in X$ is called *ϵ-far* from v if $d_X(u, v) \geq R(v, \epsilon)$. Given a function $\gamma : (0,1) \to \mathbb{R}_+$, we say that a non-contractive embedding $f : X \to Y$ has *scaling distortion* γ, if for every $\epsilon \in (0,1)$, there are at least $(1-\epsilon)\binom{|X|}{2}$ pairs that have distortion at most $\gamma(\epsilon)$. We say that f has *coarse* scaling distortion γ, if every pair $v, u \in X$ such that both u, v are $\epsilon/2$-far from each other, has distortion at most $\gamma(\epsilon)$.[10]

Moments of distortion. Let $(X, d_X), (Y, d_Y)$ be metric spaces. For $1 \leq q \leq \infty$, define the *ℓ_q-distortion* of a non-contractive embedding $f : X \to Y$ as:

$$\mathrm{dist}_q(f) = \mathbb{E}\left[\left(\frac{d_Y(f(u), f(v))}{d_G(u,v)}\right)^q\right]^{1/q},$$

where the expectation is taken according to the uniform distribution over $\binom{X}{2}$. The classic notion of *distortion* is expressed by the ℓ_∞-distortion and the *average distortion* is expressed by the ℓ_1-distortion. The following was proved in [4].

[8] The original paper claims lightness $O(\log^2(1/\epsilon))$, but their proof in fact gives the improved bound.

[9] An embedding f is non-contractive if for every $x, y \in X$, $d_Y(f(x), f(y)) \geq d_X(x, y)$.

[10] It can be verified that coarse scaling distortion γ implies scaling distortion γ.
Moreover, in this paper we prove that the definition that requires only one of u, v to be $\frac{\epsilon}{2}$-far from the other, is almost equivalent. See Remark 1.

Lemma 1. *([4]) Let (X, d_X), (Y, d_Y) be metric spaces. If a non-contractive embedding $f : X \to Y$ has scaling distortion γ then*

$$\mathrm{dist}_q(f) \le \left(2 \int_{\frac{1}{2}\binom{n}{2}^{-1}}^{1} \gamma(x)^q dx \right)^{1/q}.$$

3. Prioritized distortion vs. coarse scaling distortion

In this section we study the relationship between the notions of prioritized and scaling distortion. We show that there is a reduction that allows to transform embeddings with prioritized distortion into embeddings with coarse scaling distortion, and vice versa.

3.1. Coarse scaling distortion implies prioritized distortion

The following theorem shows that coarse scaling distortion implies prioritized distortion, implying some new prioritized distortion embedding results, and in particular a prioritized version of Bourgain's theorem.

Theorem 1. *Let $\mu : \mathbb{N} \to \mathbb{R}^+$ be a non-increasing function such that $\sum_{i \ge 1} \mu(i) = 1$. Let $\mathcal{Y}$ be a family of finite metric spaces, and assume that for every finite metric space (Z, d_Z) there exists a non-contractive embedding $f_Z : Z \to Y_Z$, where $(Y_Z, d_{Y_Z}) \in \mathcal{Y}$, with (monotone non-increasing) coarse scaling distortion γ. Then, given a finite metric space (X, d_X) and a priority ranking $x_1, \dots, x_n$ of the points of X, there exists an embedding $f : X \to Y$, for some $(Y, d_Y) \in \mathcal{Y}$, with prioritized distortion $\gamma(\mu(i))$.*

Proof. Given the metric space (X, d_X) and a priority ranking $x_1, \dots, x_n$ of the points of X, let $\delta = \min_{i \ne j} d_X(x_i, x_j)/2$. We define a new metric space (Z, d_Z) as follows. For every $1 \le i \le n$, every point x_i is replaced by a set X_i of $|X_i| = \lceil \mu(i) n \rceil$ points, and let $Z = \bigcup_{i=1}^n X_i$. For every $u \in X_i$ and $v \in X_j$ define $d_Z(u, v) = d_X(x_i, x_j)$ when $i \ne j$, and $d_Z(u, v) = \delta$ otherwise. Observe that $|Z| = \sum_{i=1}^n |X_i| \le \sum_{i=1}^n (\mu(i)n + 1) \le 2n$.

We now use the embedding $f_Z : Z \to Y_Z$ with coarse scaling distortion γ, to define an embedding $f : X \to Y_Z$, by letting for every $1 \le i \le n$, $f(x_i) = f_Z(u_i)$ for some (arbitrary) point $u_i \in X_i$. By construction of Z, for every $j > i$, we have that $X_i \subseteq B(u_i, d_Z(u_i, u_j)) \cap B(u_j, d_Z(u_i, u_j))$. As $|X_i| \ge \mu(i)n \ge \frac{\mu(i)}{2}|Z|$, it holds that u_i, u_j are $\epsilon/2$-far from each other for $\epsilon = \mu(i)$. This implies that $\frac{d_{Y_Z}(f(x_i), f(x_j))}{d_X(x_i, x_j)} = \frac{d_{Y_Z}(f_Z(u_i), f_Z(u_j))}{d_Z(u_i, u_j)} \le \gamma(\mu(i))$. $\square$

It follows from a result of [22] that the convergence condition on μ in the above theorem is necessary (more details below). We note that this reduction can also be applied to cases where the coarse scaling embedding is only known for a class of metric spaces (rather than all metrics), as long as the transformation needed for the proof can be made so that the resulting new space is still in the class. This holds for most natural classes, such as metrics arising from trees, planar graph, graphs excluding a fixed minor, bounded degree graphs, doubling metrics, etc.[11]

3.1.1. Implications

The reduction implies that all existing *coarse* scaling distortion embeddings and distance oracles have priority distortion counterparts, thus improving few of the results of [22].

Embeddings. By applying a theorem of [3] we get the following.

Corollary 2. *For every $1 \le p \le \infty$ and every finite metric space (X, d_X) and a priority ranking of X, there exists an embedding with prioritized distortion $O(\log j)$ into $l_p^{O(\log |X|)}$.*

Another consequence of the results of [3] is better bounds for decomposable metrics[12]:

Corollary 3. *For every $1 \le p \le \infty$ and every finite τ-decomposable metric space (X, d_X) and a priority ranking of X, there exists an embedding with prioritized distortion $O(\tau^{1-1/p}(\log j)^{1/p})$ into $l_p^{O(\log^2 |X|)}$.*

---

[11] For the graph classes mention above (as well as for doubling metrics), a small change in the construction is needed. From each original vertex x_i, we will grow a path X_i of $\lceil \mu(i)n \rceil$ vertices, where all the path edges have weight $\frac{\delta \cdot \alpha}{2n}$ for arbitrarily small $\alpha > 0$. We will also need to choose u_i to be the single leaf in the added path (rather then simply arbitrarily chosen vertex). The same proof will guarantee a $(1+\alpha)\gamma(\mu(i))$ prioritized distortion.

[12] Roughly, a metric space is called τ-decomposable if it allows probabilistic partitions with padding parameter τ; e.g. Planar metrics and doubling metrics. An exact definition appears in [3].

Spanners. Applying Theorem 1 on [16, Corollary 3] we get a linear size prioritized spanner.

Corollary 4. *Given a graph $G = (V, E)$ and any priority ranking $v_1, v_2, \dots, v_n$ of V, there exists a spanner H with $O(n)$ edges and prioritized distortion $O(\log j)$.*

We remark that in Theorem 6 we show directly a spanner with $O(n \log\log n)$ edges and prioritized distortion $\tilde{O}(\log j)$ (which could easily be made $O(\log j)$). While not being of linear size, that spanner is very light. We currently do not know how to achieve both lightness and linear size spanner with prioritized distortion $O(\log j)$.

Distance oracles. In [22], among other possible tradeoffs, it was shown how to construct distance oracles with $O(n \log\log n)$ space and prioritized distortion $O(\log n / \log(n/j))$ with $O(1)$ query time. (Alternatively, they had $O(n \log^* n)$ space and $O(2^{\log n/\log(n/j)})$ prioritized stretch with $O(1)$ query time.) The space requirement in [22] was never truly linear in n. Chechick [18] showed that for every metric space (X, d_X), one can construct a distance oracle with $O(\log n)$-stretch, $O(1)$-query time and $O(n)$ space. A black box reduction from [16], will provide us with distance oracle with $O(\log \frac{1}{\epsilon})$ coarse scaling distortion, $O(1)$-query time and $O(n)$ space. We conclude with a linear size prioritized distance oracle.

Corollary 5. *For every metric space (X, d_x) and every priority ranking, there exist a distance oracle requiring $O(n)$ space, that answer distance queries in $O(1)$ time and $O(\log j)$ priority distortion.*

We remark that the prioritized distortion $O(\log n/\log(n/j))$ of [22] is superior to our $O(\log j)$.

3.2. Prioritized distortion implies coarse scaling distortion

Here we prove the direction that is used for our main result of a light constant average distortion spanning tree, specifically, that prioritized distortion implies scaling distortion.

Theorem 2. *Let (X, d_X), (Y, d_Y) be metric spaces, then there exists a priority ranking $\pi = x_1, \dots, x_n$ of the points of X such that the following holds: If there exists an embedding $f : X \to Y$ with (monotone non-decreasing) prioritized distortion α (with respect to π), then f has coarse scaling distortion $O(\alpha(8/\epsilon))$.*

The basic idea of the proof is to choose the priorities so that for every ϵ, every $v \in X$ has a representative v' of sufficiently high priority within distance $\approx R(v, \epsilon)$. Then for any $u \in X$ which is ϵ-far from v, we can use the low distortion guarantee of v' with both v and u via the triangle inequality. To this end, we employ the notion of a *density net* due to [16], who showed that a greedy construction provides such a net.

Definition 1 *(Density net).* Given a metric space (X, d) and a parameter $0 < \epsilon < 1$, an ϵ*-density-net* is a set $N \subseteq X$ such that: **1)** for all $v \in X$ there exists $u \in N$ with $d(v, u) \le 2R(v, \epsilon)$ and **2)** $|N| \le \frac{1}{\epsilon}$.

Proof of Theorem 2. We begin by describing π, the desired priority ranking of X. For every integer $1 \le i \le \lceil \log n \rceil$ let $\epsilon_i = 2^{-i}$, and let $N_i \subseteq X$ be an ϵ_i-density-net in X. Set π to be a priority ranking of X satisfying that every point $v \in N_i$ has priority at most $\left|\bigcup_{j=1}^{i} N_j\right| \le \sum_{j=1}^{i} |N_j|$. As for any j, $|N_j| \le \frac{1}{\epsilon_j} = 2^j$, each point in N_i has priority at most $\sum_{j=1}^{i} \frac{1}{\epsilon_j} \le \sum_{j=1}^{i} 2^j < 2^{i+1}$.

Let $f : X \to Y$ be some non-contractive embedding with priority distortion α with respect to π. Fix some $\epsilon \in (0, 1)$ and a pair $v, u \in V$ so that u is ϵ-far from v. Let i be the minimal integer such that $\epsilon_i \le \epsilon$ (note that we may assume $1 \le i \le \lceil \log n \rceil$, because there is nothing to prove for $\epsilon < 1/n$). By Definition 1 we can take $v' \in N_i$ such that $d(v, v') \le 2R(v, \epsilon_i)$. As u is ϵ-far from v, it holds that

$$d_X(v, v') \le 2R(v, \epsilon_i) \le 2R(v, \epsilon) \le 2d_X(v, u) . \tag{1}$$

In particular, by the triangle inequality,

$$d_X(u, v') \le d_X(u, v) + d_X(v, v') \overset{(1)}{\le} 3d_X(u, v) . \tag{2}$$

The priority of v' is at most 2^{i+1}, hence

$$\begin{aligned} & d_Y(f(v), f(u)) \\ & \quad \le \quad d_Y(f(v), f(v')) + d_Y(f(v'), f(u)) \\ & \quad \le \quad \alpha(2^{i+1}) \cdot d_X(v, v') + \alpha(2^{i+1}) \cdot d_X(v', u) \\ & \quad \overset{(1)\wedge(2)}{\le} \quad 5\alpha(2/\epsilon_i) \cdot d_X(v, u) . \end{aligned}$$

By the minimality of i it follows that $1/\epsilon_i \le 2/\epsilon$, and since α is monotone

$$d_Y(f(v), f(u)) \le 5\alpha(2/\epsilon_i) \cdot d_X(v, u) \le 5\alpha(4/\epsilon) \cdot d_X(v, u) \ ,$$

as required. Since we desire distortion guarantee for pairs that are $\epsilon/2$-far, the distortion becomes $O(\alpha(8/\epsilon))$. □

Remark 1. The proof of Theorem 2 provides an even stronger conclusion, that any pair $u, v \in X$ such that one is $\epsilon/2$-far from the other, has the claimed distortion bound. While in the original definition of coarse scaling both points are required to be $\epsilon/2$-far from each other, it is often the case that we achieve the stronger property. Yet, in some of the cases in previous work the weaker definition seemed to be of importance. Combining Theorem 2 and Theorem 1, we infer that essentially any coarse scaling embedding can have such a one-sided guarantee, with a slightly worse dependence on ϵ, as claimed in the following corollary.

Corollary 6. *Let $\mu : \mathbb{N} \to \mathbb{R}^+$ be a non-increasing function such that $\sum_{i \ge 1} \mu(i) = 1$. Let $\mathcal{Y}$ be a family of finite metric spaces, and assume that for every finite metric space (Z, d_Z) there exists a non-contractive embedding $f_Z : Z \to Y_Z$, where $(Y_Z, d_{Y_Z}) \in \mathcal{Y}$, with (monotone non-increasing) coarse scaling distortion $\gamma(\epsilon)$. Then given any finite metric space X, there exists an embedding $f : X \to Y$, for some $(Y, d_Y) \in \mathcal{Y}$, with (monotone non-decreasing)* one-sided *coarse scaling distortion $O(\gamma(\mu(8/\epsilon)))$.*

Proof. By the assumption, there exists $(Y, d_Y) \in \mathcal{Y}$ so that X embeds to Y with coarse scaling distortion $\gamma(\epsilon)$. According to Theorem 1, there is an embedding f with prioritized distortion $\gamma(\mu(i)))$ (w.r.t. to any fixed priority ranking π). We pick π to be the ordering required by Theorem 2, and conclude that f has one-sided coarse scaling distortion $O(\gamma(\mu(8/\epsilon)))$. □

3.3. Coarse partial distortion and terminal distortion

As a special case of the reductions Theorem 1 and Theorem 2, we can prove an equivalence between coarse partial distortion to terminal distortion.

Definition 2 *(Coarse partial distortion).* Let $(X, d_X), (Y, d_Y)$ be metric spaces, and let $\epsilon \in (0, 1)$, $\gamma \ge 1$. A non-contractive embedding $f : X \to Y$ has $(1 - \epsilon)$-coarse partial scaling distortion γ, if every pair $v, u \in X$ such that both u, v are $\epsilon/2$-far from each other, has distortion at most γ.

Note that the embedding f has coarse scaling distortion γ if and only if for every $\epsilon \in (0, 1)$, f has $(1 - \epsilon)$-coarse partial distortion $\gamma(\epsilon)$.

Definition 3 *(Terminal distortion).* Let $(X, d_X), (Y, d_Y)$ be metric spaces, and $K \subseteq X$ a subset of terminals. A non-contractive embedding $f : X \to Y$ has terminal distortion α w.r.t. K, if every pair $(v, u) \in K \times X$ has distortion at most α.

Note that for a priority ranking $\pi = x_1, \ldots, x_n$, the embedding f has priority distortion α w.r.t. π if and only if for every k, f has terminal distortion $\alpha(k)$ w.r.t. $K = \{x_1, \ldots, x_k\}$. It is important to note that Definition 3 differs from the original definition of terminal distortion in [21], which did not require f to be non-contractive on *all pairs*. We elaborate on this issue in Subsection 3.3.1.

Theorem 3. *Let $\mu : \mathbb{N} \to \mathbb{R}^+$ be a non-increasing function such that $\sum_{i \ge 1} \mu(i) = 1$. Let $k \in \mathbb{N}$. Let $\mathcal{Y}$ be a family of finite metric spaces, and assume that for every finite metric space (Z, d_Z) there exists an embedding $\phi : Z \to Y_Z$, where $(Y_Z, d_{Y_Z}) \in \mathcal{Y}$, with coarse $(1 - 1/(2k))$-partial distortion γ. Then, given a finite metric space (X, d_X) and a set of terminals $K \subsetneq X$ of size $|K| = k$, there exists an embedding $f : X \to Y$, for some $(Y, d_Y) \in \mathcal{Y}$, with terminal distortion γ.*

Proof. Simply follow the proof of Theorem 1, using $\mu(x) = \frac{1}{2k}$ for $x \in K$, and $\mu(x) = \frac{1}{2(|X| - k)}$ for $x \in X \setminus K$. As every $x \in K$ has $\frac{n}{2k}$ copies, and the new metric Z contains at most $2n$ points, x is $\frac{1}{4k}$-far from any other $y \in X$ (in the metric space Z). Also this y is $\frac{1}{4k}$-far from x (since $|B_Z(y, d(x, y))| \ge n/(2k)$). Thus the embedding with coarse $(1 - 1/(2k))$-partial distortion for Z has distortion at most γ for such a pair x, y. □

Theorem 4. *Let (X, d_X), (Y, d_Y) be metric spaces, and $k \in \mathbb{N}$ a parameter. There exists a subset $K \subseteq X$ of size k, such that the following holds: If there exists an embedding $f : X \to Y$ with terminal distortion α, then f has coarse $(1 - \frac{2}{k})$-partial scaling distortion 5α.*

Proof. Following the lines of the proof of Theorem 2 let K be a $\frac{1}{k}$-density net. Fix a pair $v, u \in V$ so that u is $\frac{1}{2} \cdot \frac{2}{k} = \frac{1}{k}$-far from v. Let $v' \in N$ such that $d_X(v, v') \le 2R(v, \frac{1}{k}) \le 2d_X(v, u)$. It holds that,

$$\begin{aligned} d_Y(f(v), f(u)) &\le d_Y(f(v), f(v')) + d_Y(f(v'), f(u)) \\ &\le \alpha \cdot d_X(v, v') + \alpha \cdot d_X(v', u) \\ &\le 2\alpha \cdot d_X(v, u) + 3\alpha \cdot d_X(v, u) = 5\alpha \cdot d_X(v, u) . \quad \square \end{aligned}$$

Among other implications, Theorem 4 implies the following:

Corollary 7. *For every parameters $0 < \epsilon, \delta < 1$, every n-vertex weighted graph G contains a spanning tree with $(1-\epsilon)$-coarse partial distortion $O(\frac{1}{\epsilon\delta})$ and $1+\delta$ lightness.*

Proof. In [21] it was shown that for every weighted graph $G = (V, E, w)$ and terminal set $K \subseteq V$ of size k, there is a spanning tree T with terminal distortion $O(k)$ and constant lightness. Using Theorem 8 (proven below), we get that G contains a spanning tree with terminal distortion $O(k/\delta)$ and lightness $1+\delta$. Now, Theorem 4 (with $k = \frac{2}{\epsilon}$) implies the corollary. $\square$

3.3.1. Weak terminal distortion

Our definition of terminal distortion has a one-sided guarantee on *all pairs*, e.g., the embedding must not contract any distance. This definition differs from the original definition of terminal distortion which appears in [21], where the non-contractive requirement was missing (formally, by [21], f has terminal distortion α iff there is some constant $c \in \mathbb{R}$ such that $\forall (v, u) \in K \times X$, $d_X(u, v) \le c \cdot d_Y(f(u), f(v)) \le \alpha \cdot d_X(u, v)$). We will refer to the original definition from [21] as *weak terminal distortion*.

These two definitions are indeed different. For example, in [21] it was shown that given n points containing k terminals in $\mathbb{R}^n$, they can be embedded into $\mathbb{R}^{O(\log k)}$ with weak terminal distortion $O(1)$ (under the ℓ_2-norm). However, any non-contracting embedding with constant distortion requires $\Omega(\log n)$ dimensions, so this is impossible under our Definition 3. As a result of the difference between these definitions, there are some results in [21] on which the reduction of Theorem 4 cannot be used.

Nevertheless, if the target space is ℓ_p, we devise a transformation from weak terminal distortion into terminal distortion, while increasing the dimension *additively* by $O(\log n)$. The first step is Theorem 5, in which we extend a standard embedding into a terminal one, in a different manner than [21]. This theorem has other implications: in particular, we generalize and improve the dimension in a result of [1,3] on embedding into ℓ_p with coarse partial distortion.

Theorem 5. *Let (X, d_X) be metric space of size n, and $K \subseteq X$ be a subset of terminals. Suppose that there exists an embedding $f : K \to \ell_p^\beta$ with distortion α, then there is an embedding $\hat{f} : X \to \ell_p^{\beta+O(\log n)}$ with terminal distortion $O(\alpha)$.*

Proof. Assume, as we may, that f is non-contractive. That is, for every $v, u \in K$, $d_X(u, v) \le \|f(u) - f(v)\|_p \le \alpha \cdot d_X(u, v)$.

Fix $m = O(\log n)$. Let $g : X \to \{\pm 1\}^m$ such that for every $v, u \in X$, there are at least $\frac{m}{4}$ coordinates i where $g_i(v) \neq g_i(u)$ (a random g will work with high probability, as can be verified by Chernoff inequality). For every vertex $u \in X$, let k_u be the closest terminal to u. The embedding $\hat{f}$ is defined as follows. For $u \in X$,

$$\hat{f}(u) = f(k_u) \oplus \frac{d_X(u, k_u)}{m^{1/p}} \cdot g(u) .$$

First, we will show that $\hat{f}$ has expansion at most $O(\alpha)$ on terminal pairs. Fix some $v \in K$ and $u \in X$.

$$\begin{aligned} \|\hat{f}(v) - \hat{f}(u)\|_p^p &= \|f(v) - f(k_u)\|_p^p + \frac{1}{m}\sum_{i=1}^{m} |g_i(u) \cdot d_X(u, k_u)|^p \\ &\le \alpha^p \cdot d_X(v, k_u)^p + d_X(u, k_u)^p \\ &\le \left(\alpha^p + 1\right) \cdot (d_X(v, u) + d_X(u, k_u))^p \\ &\le \left(\alpha^p + 1\right) \cdot (2 d_X(v, u))^p . \end{aligned}$$

Thus, $\|\hat{f}(v) - \hat{f}(u)\|_p \le 2\left(\alpha^p + 1\right)^{1/p} \cdot d_X(v, u) \le 2(\alpha + 1) \cdot d_X(v, u)$.

Next, we bound the contraction for all pairs. Fix some $v, u \in X$. If $d_X(u, v)/2 \ge d_X(v, k_v) + d_X(u, k_u)$, then

$$\begin{aligned} \|\hat{f}(v) - \hat{f}(u)\|_p &\ge \|f(k_v) - f(k_u)\|_p \ge d_X(k_v, k_u) \\ &\ge d_X(v, u) - d_X(v, k_v) - d_X(u, k_u) \ge d_X(v, u)/2 . \end{aligned}$$

Otherwise,

$$\|\hat{f}(v)-\hat{f}(u)\|_p^p \geq \frac{1}{m}\sum_{i=1}^{m}|g_i(v)\cdot d_X(v,k_v)-g_i(u)\cdot d_X(u,k_u)|^p$$
$$=\frac{1}{m}\cdot\frac{m}{4}\cdot|d_X(v,k_v)+d_X(u,k_u)|^p \geq \frac{1}{4}\cdot(d_X(v,u)/2)^p \ .$$

To ensure the embedding does not contract, our final embedding will be $2^{1+\frac{2}{p}}\cdot\hat{f}$. □

We now show the transformation from weak terminal distortion to (non-contracting) terminal distortion. (By Theorem 4 this can provide embeddings with coarse partial distortion as well.) Suppose an embedding $f: X\to Y$ has weak terminal distortion α. In particular, its restriction to K has distortion α. Using Theorem 5 we conclude:

Corollary 8. *Let (X,d_X) be metric space of size n, and $K\subseteq X$ be a subset of terminals. Suppose that there exists an embedding $f: X\to\ell_p^\beta$ with weak terminal distortion α, then there exist embedding $\hat{f}: X\to\ell_p^{\beta+O(\log n)}$ with terminal distortion $O(\alpha)$.*

Fix some $p\geq 1$. Let $\mathcal{X}$ be a subset-closed family of finite metric spaces such that for any $n\geq 1$ and any n-point metric space $X\in\mathcal{X}$ there exists an embedding $f_X: X\to\ell_p$ with distortion $\alpha(n)$ and dimension $\beta(n)$. In [1,3] it was shown that, assuming all f_X are strongly non-expansive,[13] there is a universal constant C and an embedding from X into ℓ_p with $(1-\epsilon)$-coarse partial distortion $O(\alpha(C/\epsilon))$ and dimension $\beta(C/\epsilon)\cdot O(\log n)$. By combining Theorem 5 with Theorem 4 we considerably improve the dimension, and remove the strongly non-expansive requirement.

Corollary 9. *Fix some $p\geq 1$. Let $\mathcal{X}$ be a subset-closed family of finite metric spaces such that for any $n\geq 1$ and any n-point metric space $X\in\mathcal{X}$ there exists an embedding $f_X: X\to\ell_p$ with distortion $\alpha(n)$ and dimension $\beta(n)$. Then there is an embedding from X into ℓ_p with $(1-\epsilon)$-coarse partial distortion $O(\alpha(2/\epsilon))$ and dimension $\beta(2/\epsilon)+O(\log n)$.*

4. Light spanner with prioritized distortion

In this section we prove that every graph admits a light spanner with bounded prioritized distortion. (The runtime analysis of our algorithms appears in Subsection 4.2.)

Theorem 6 *(Prioritized spanner). Given a graph $G=(V,E)$, a parameter $0<\rho<1$ and any priority ranking $v_1,v_2,\ldots,v_n$ of V, there exists a spanner H with lightness $1+\rho$ and prioritized distortion $\tilde{O}(\log j)/\rho$.*

Combining Theorem 6 and Theorem 2 we obtain the following.

Theorem 7. *For any parameter $0<\rho<1$, any graph contains a spanner with coarse scaling distortion $\tilde{O}(\log(1/\epsilon))/\rho$ and lightness $1+\rho$. Moreover, the spanner can be computed in $O(n\cdot(m+n\log n))$ time.*

By Lemma 1 it follows that this spanner has ℓ_q-distortion $\tilde{O}(q)/\rho$ for any $1\leq q<\infty$.

We can also obtain a spanner with both scaling distortion and prioritized distortion simultaneously, where the priority is with respect to an arbitrary ranking $\pi=v_1,\ldots,v_n$. To achieve this, one may define a ranking which interleaves π with the ranking generated in the proof of Theorem 2.

We now turn to proving Theorem 6. The proof is based on the following main technical lemma:

Lemma 10. *Given a graph $G=(V,E)$, a subset $K\subseteq V$ of size k, and a parameter $0<\delta<1$, there exists a spanner H that **1)** contains the MST of G, **2)** has lightness $1+\delta$, and **3)** every pair in $K\times V$ has distortion $O((\log k)/\delta)$.*

Before proving this lemma, let us first apply it to prove Theorem 6.

Proof of Theorem 6. For every $1\leq i\leq\lceil\log\log n\rceil$ let $K_i=\left\{v_j\ :\ j\leq 2^{2^i}\right\}$. Let H_i be the spanner given by Lemma 10 with respect to the set K_i and the parameter $\delta_i=\rho/i^2$. Hence H_i has $1+\rho/i^2$ lightness and $O\left(\frac{\log|K_i|}{\delta_i}\right)=O(2^i\cdot i/\rho)$ distortion for pairs in $K_i\times V$. Let $H=\bigcup_i H_i$ be the union of all these spanners (that is, the graph containing every edge of every one of these spanners). As each H_i contains the unique MST of G, it holds that

[13] Embedding $f: X\to\ell_p$ is strongly non-expansive if $f=(\eta_1 f_1,\ldots,\eta_m f_m)$ where $\sum_{i=1}^m\eta_i=1$, and each f_i is non-expansive embedding into $\mathbb{R}$.

$$\Psi(H) \leq 1 + \sum_{i \geq 1} \rho / i^2 = 1 + O(\rho) \ .$$

To see the prioritized distortion, let $v_j, v_r \in V$ be such that $j < r$, and let $1 \leq i \leq \lceil \log \log n \rceil$ be the minimal index such that $v_j \in K_i$. Note that $2^{2^{i-1}} \leq j$, and in particular $2^{i-1} \leq \log j$ (with the exception of $j = 1$, but that case holds by the virtue of $j = 2$, say). This implies that

$$\begin{aligned} d_H(v_j, v_r) \leq d_{H_i}(v_j, v_r) &\leq O(2^i \cdot i^2 / \rho) \cdot d_G(v_j, v_r) \\ &\leq \tilde{O}(\log j) / \rho \cdot d_G(v_j, v_r) \ , \end{aligned}$$

as required. □

4.1. Proof of Lemma 10

The construction of the spanner that fulfills the properties promised in Lemma 10 is as follows. First, we use the spanner of [19] to get a spanner with lightness $O(1)$ and distortion $O(\log k)$ over pairs in $K \times K$. Then, by combining this spanner with the SLT by [27], we expand the $O(\log k)$ distortion guarantee to all pairs in $K \times V$, while the lightness is still $O(1)$. Finally, we use a general reduction (Theorem 8), that reduces the weight of a spanner while increasing its distortion. By applying the reduction, we get a spanner with $1 + \rho$ lightness while paying additional factor of $1/\rho$ in the distortion.

We begin by describing the general reduction.

Theorem 8. *Let $G = (V, E)$ be a graph, $0 < \delta < 1$ a parameter and $t : \binom{V}{2} \to \mathbb{R}_+$ some function. Suppose that for every weight function $w : E \to \mathbb{R}_+$ there exists a spanner H with lightness ℓ such that every pair $u, v \in V$ suffers distortion at most $t(u, v)$. Then for every weight function w there exists a spanner H with lightness $1 + \delta\ell$ and such that every pair u, v suffers distortion at most $t(u, v)/\delta$. Moreover, H contains the MST of G with respect to w.*

Proof. Fix some weight function w and let $G = (V, E, w)$ be the graph associated with this weight function, and let T be the MST of G. Set $w' : E \to \mathbb{R}_+$ to be a new weight function

$$w'(e) = \begin{cases} w(e) & e \in T \\ w(e)/\delta & e \notin T \end{cases} ,$$

that is, we multiply the weight of all non-MST edges by $1/\delta$. Let $G' = (V, E, w')$ be the graph G associated with the new weight function w'. Note that T is also the MST of G' (since the weight of any spanning tree is higher in G' than in G except for T itself). By our assumption there exists a spanner $H' = \left(V, E_{H'}, w'\right)$ of G' with distortion bounded by t and lightness ℓ. Set $H = (V, E_{H'} \cup T, w)$ as a spanner of G. The edge set of H consists of the edges of H' together with the MST edges, all with the original weight function w.

As the weight of the non-MST edges are larger in G' by $1/\delta$ factor compared to their weight in G, we have

$$\begin{aligned} w(E_H) = w(T) + w\left(E_{H'} \setminus T\right) &= w(T) + \delta \cdot w'\left(E_{H'} \setminus T\right) \leq w(T) + \delta \cdot w'\left(E_{H'}\right) \\ &\leq w(T) + \delta\ell \cdot w'(T) = (1 + \delta\ell) \cdot w(T) \ , \end{aligned}$$

concluding that the lightness of H is at most $1 + \delta\ell$.

To bound the distortion, consider an arbitrary pair of vertices $u, v \in V$. Let $P_{u,v}$ be the shortest path from u to v in G. As for each edge $e \in P_{u,v}$, $w'(e) \leq w(e)/\delta$ we have that

$$d_{G'}(u, v) \leq \sum_{e \in P_{u,v}} w'(e) \leq \sum_{e \in P_{u,v}} \frac{1}{\delta} \cdot w(e) = \frac{1}{\delta} \cdot d_G(u, v) \ .$$

Therefore:

$$d_H(u, v) \leq d_{H'}(u, v) \leq t(u, v) \cdot d_{G'}(u, v) \leq \frac{t(u, v)}{\delta} d_G(u, v) \ ,$$

as required. □

In a recent work, Chechik and Wulff-Nilsen achieved the following result:

Theorem 9 *([19]). For every weighted graph $G = (V, E, w)$ and parameters $k \geq 1$ and $0 < \epsilon \leq 1$, there exist a polynomial time algorithm that constructs a spanner with distortion $(2t - 1)(1 + \epsilon)$ and lightness $n^{1/t} \cdot \mathrm{poly}(\frac{1}{\epsilon})$.*

Note that for an n-vertex graph with parameters $t = \log n, \epsilon = 1$, they get a spanner with distortion $O(\log n)$ and constant lightness. However, their construction does not seem to provide lightness arbitrarily close to 1.

A tree $\mathcal{T} = (V', E', w')$ is called a *Steiner tree* for a graph $G = (V, E, w)$ if (1) $V \subseteq V'$, and (2) for any pair of vertices $u, v \in V$ it holds that $d_{\mathcal{T}}(u, v) \geq d_G(u, v)$. The *minimum Steiner tree* T of G, denoted $SMT(G)$, is a Steiner tree of G with minimum weight. It is well-known that for any graph G, $w(SMT(G)) \geq \frac{1}{2}MST(G)$. (See, e.g., [26], Section 10.)

We will use [19] spanner to construct a spanner with $O(1)$ lightness and distortion $O(\log k)$ over pairs in $K \times K$. Let $G_k = (K, \binom{K}{2}, w_k)$ be the complete graph over the terminal set K with weights $w_k(u, v) = d_G(u, v)$ (for $u, v \in K$) that are given by the shortest path metric in G. Let T_k be the MST of G_k. Note that the MST T of G is a Steiner tree of G_k, hence $w_k(T_k) \leq 2 \cdot w_k(SMT(G_k)) \leq 2 \cdot w(T)$.

Using Theorem 9, let $H_k = (K, E_k, w_k)$ be a spanner of G_k with weight $O(w_k(T_k)) = O(w(T))$ (constant lightness) and distortion $O(\log k)$. For a pair of vertices $u, v \in K$, let P_{uv} denote the shortest path between u and v in G. Let $H' = (V, E', w)$ be a subgraph of G with the set of edges $E' = \cup_{\{u,v\}\in E_k} P_{uv}$ (i.e. for every edge $\{u, v\}$ in H_k, we take the shortest path from u to v in G). It holds that,

$$w(H') \leq \sum_{\{u,v\}\in E_k} w(P_{uv}) = \sum_{e\in E_k} w_k(e) = O(w(T)) .$$

Moreover, for every pair $u, v \in K$,

$$d_{H'}(u, v) \leq d_{H_k}(u, v) \leq O(\log k) \cdot d_{G_k}(u, v) = O(\log k) \cdot d_G(u, v) . \tag{3}$$

Now we extend H' so that every pair in $K \times V$ will suffer distortion at most $O(\log k)$. To this end, we use the following lemma regarding shallow light trees (SLT), which is implicitly proved in [27,6].

Lemma 11. *Given a graph $G = (V, E)$, a parameter $\alpha > 1$, and a subset $K \subseteq V$, there exists a spanner S[14] of G with lightness $1 + \frac{2}{\alpha-1}$, and for any vertex $u \in V$, $d_S(u, K) \leq \alpha \cdot d_G(u, K)$.*

Let S be the spanner of Lemma 11 with respect to the set K and parameter $\alpha = 2$. Define H'' as the union of H' and S. As both H' and S have constant lightness, so does H''. It remains to bound the distortion of an arbitrary pair $v \in K$ and $u \in V$. Let $k_u \in K$ be the closest vertex to u among the vertices in K with respect to the distances in the spanner S. By the assertion of Lemma 11,

$$d_S(u, k_u) = d_S(u, K) \leq 2 \cdot d_G(u, K) \leq 2 \cdot d_G(u, v) . \tag{4}$$

Using the triangle inequality,

$$d_G(v, k_u) \leq d_G(v, u) + d_G(u, k_u) \leq d_G(v, u) + d_S(u, k_u) \overset{(4)}{\leq} 3 \cdot d_G(v, u) . \tag{5}$$

Since both $v, k_u \in K$ it follows that

$$d_{H'}(v, k_u) \overset{(3)}{\leq} O(\log k) \cdot d_G(v, k_u) \overset{(5)}{\leq} O(\log k) \cdot d_G(v, u) . \tag{6}$$

We conclude that

$$d_{H''}(v, u) \leq d_{H'}(v, k_u) + d_S(k_u, u) \overset{(4)\wedge(6)}{\leq} O(\log k) \cdot d_G(v, u) .$$

We showed a polynomial time algorithm, that given a weighted graph $G = (V, E, w)$ and a subset $K \subseteq V$ of size k, constructs a spanner H with lightness $O(1)$, and such that every pair in $K \times V$ has distortion at most $O(\log k)$. Now Theorem 8 implies Lemma 10.

4.2. Efficient implementations

First we describe an $O(k \cdot (m + n\log n))$-time algorithm for the terminal spanner of Lemma 10. We start by constructing G_k, the full graph on K. For this goal we need to compute all distances in the terminal set K. This can be done by computing k shortest path trees rooted in each terminal vertex in K, using Dijkstra's algorithm it will take $O(k \cdot (m + n\log n))$ time. Our next step is to compute a light spanner for G_k. Instead of using [19] in the construction of the spanner H_k, we will use a more efficient construction by Alstrup et al. [7], who provide us with a $O(\log k)$ stretch spanner with $O(1)$ lightness and $O(k)$ edges in $O(k^2)$ time.[15] Next, for every edge $\{u, v\} \in H_k$ we find the shortest path between u and v in G and add it

[14] In fact, S is a spanning forest of G.

[15] See Theorem 4 in [7]. In fact, for n-vertex graph with m edges with stretch parameter $O(\log n)$, their running time is $O(m + n^{1+\epsilon'})$ for arbitrarily small constant ϵ'.

to H'. This is done using the shortest path tree rooted in u computed earlier in $O(n)$ time per edge. Thus H' is constructed from H_k in total $O(k \cdot n)$ time. Our next step is to compute an SLT rooted in K using the $O(m + n \log n)$ time algorithm of [27]. Finally, it is straightforward that the spanner reduction lemma takes linear time (once the MST is computed). We conclude that the total running time of the spanner construction of Lemma 10 is $O(k \cdot (m + n \log n))$.

Next, in order to compute the prioritized spanner of Theorem 6 we simply construct terminal spanners for the sets K_i, $1 \le i \le \lceil \log \log n \rceil$. The total time required is $\sum_{i=1}^{\lceil \log \log n \rceil} O\left(|K_i| \cdot (m + n \log n)\right) = O\left(n \cdot (m + n \log n)\right)$.

Finally, we analyze the running time of Theorem 7. Theorem 7 is achieved by combining Theorem 6 with Theorem 2. First we need to compute the priority ranking of Theorem 6, which is based on density nets. Specifically we need to compute for every $1 \le i \le \lceil \log n \rceil$, a 2^{-i} density net. [16] did not analyze explicitly the running time required for computing a density net, but it is not hard to see that the running time of their algorithm is $O(n \cdot (m + n \log n))$ (the time it takes to compute all-pairs-shortest-paths).

5. A light tree with constant average distortion

Here we prove our main theorem on finding a light spanning tree with constant average distortion. Later on we show a probabilistic embedding into a distribution of light spanning trees with improved bound on higher moments of the distortion.

Theorem 10. *For any parameter $0 < \rho < 1$, any graph contains a spanning tree with scaling distortion $\tilde{O}(\sqrt{1/\epsilon})/\rho$ and lightness $1 + \rho$. Moreover, this tree can be found in $\tilde{O}(m \cdot n)$ time.*

It follows from Lemma 1 that the average distortion of the spanning tree obtained is $O(1/\rho)$. Moreover, the ℓ_q-distortion is $O(1/\rho)$ for any fixed $1 \le q < 2$, $\tilde{O}\left(\log^{1.5} n\right)/\rho$ for $q = 2$, and $\tilde{O}(n^{1-2/q})/\rho$ for any fixed $2 < q < \infty$.

We will need the following simple lemma, that asserts the scaling distortion of a composition of two maps is essentially the product of the scaling distortions of these maps.[16]

Lemma 12. *Let (X, d_X), (Y, d_Y) and (Z, d_Z) be metric spaces. Let $f : X \to Y$ (respectively, $g : Y \to Z$) be a non-contractive onto embedding with scaling distortion α (resp., β). Then $g \circ f$ has scaling distortion $\alpha(\epsilon/2) \cdot \beta(\epsilon/2)$.*

Proof. Let $n = |X|$. Let $\text{dist}_f(v, u) = \frac{d_Y(f(v), f(u))}{d_X(v,u)}$ be the distortion of the pair $u, v \in X$ under f, and similarly let $\text{dist}_g(v, u) = \frac{d_Z(g(f(v)), g(f(u)))}{d_Y(f(v), f(u))}$. Fix some $\epsilon \in (0, 1)$. We would like to show that at most $\epsilon \cdot \binom{n}{2}$ pairs suffer distortion greater than $\alpha(\epsilon/2) \cdot \beta(\epsilon/2)$ by $g \circ f$. Let $A = \left\{\{v, u\} \in \binom{X}{2} : \text{dist}_f(v, u) > \alpha(\epsilon/2)\right\}$ and $B = \left\{\{v, u\} \in \binom{X}{2} : \text{dist}_g(v, u) > \beta(\epsilon/2)\right\}$. By the bound on the scaling distortions of f and g, it holds that $|A \cup B| \le |A| + |B| \le \epsilon \cdot \binom{n}{2}$. Note that if $\{v, u\} \notin A \cup B$ then

$$\begin{aligned}\frac{d_Z\left(g(f(v)), g(f(u))\right)}{d_X(v, u)} &= \text{dist}_f(v, u) \cdot \text{dist}_g(v, u) \\ &\le \alpha(\epsilon/2) \cdot \beta(\epsilon/2)\,,\end{aligned}$$

which concludes the proof. □

We will also need the following result, that was proved in [4]. (The bound on the running time was not explicitly stated in [4]. Their algorithm uses the star-decomposition of [20] whose running time is $\tilde{O}(m)$, and a certain iterative algorithm that chooses a radius for each of the $\tilde{O}(n)$ cones, which can be trivially implemented in $\tilde{O}(n)$ time.)

Theorem 11 *([4]). Any graph contains a spanning tree with scaling distortion $O(\sqrt{1/\epsilon})$, and this tree can be found in $\tilde{O}(m \cdot n)$ time.*

Now we can prove the main result.

Proof of Theorem 10. Let H be the spanner given by Theorem 7. Let T be a spanning tree of H constructed according to Theorem 11. By Lemma 12, T has scaling distortion $O(\sqrt{1/\epsilon}) \cdot \tilde{O}(\log(1/\epsilon))/\rho = \tilde{O}(\sqrt{1/\epsilon})/\rho$ with respect to the distances in G. The lightness follows as $\Psi(T) \le \Psi(H) \le 1 + \rho$. □

Random tree embedding. We also derive a result on probabilistic embedding into light spanning trees with scaling distortion. That is, the embedding construct a distribution over spanning tree so that each tree in the support of the distribution

[16] Note that this is not true for the average distortion – one may compose two maps with constant average distortion and obtain a map with $\Omega(n)$ average distortion.

is light. In such probabilistic embeddings [10] into a family $\mathcal{Y}$, each embedding $f = f_Y : X \to Y$ (for some $(Y, d_Y) \in \mathcal{Y}$) in the support of the distribution is non-contractive, and the distortion of the pair $u, v \in X$ is defined as $\mathbb{E}_Y\left[\frac{d_Y(f(u),f(v))}{d_X(u,v)}\right]$. The prioritized and scaling distortions are defined accordingly. We make use of the following result from [4].[17]

Theorem 12. *([4]) Every weighted graph G embeds into a distribution over spanning trees with coarse scaling distortion* $\tilde{O}(\log^2(1/\epsilon))$.

We note that the distortion bound on the composition of maps in Lemma 12 also holds whenever g is a random embedding, and we measure the scaling expected distortion. Thus, following the same lines as in the proof of Theorem 10, (while using Theorem 12 instead of Theorem 11), we obtain the following.

Theorem 13. *For any parameter* $0 < \rho < 1$ *and any weighted graph G, there is an embedding of G into a distribution over spanning trees with scaling distortion* $\tilde{O}(\log^3(1/\epsilon))/\rho$*, such that every tree T in the support has lightness* $1 + \rho$.

It follows from Lemma 1 that the ℓ_q-distortion is $O(1/\rho)$, for every fixed $q \geq 1$.

6. Lower bound on the trade-off between lightness and average distortion

In this section, we give an example of a graph for which any spanner with lightness $1+\rho$ has average distortion $\Omega(1/\rho)$ (of course this bound holds for the ℓ_q-distortion as well). This shows that our results are tight.[18]

Lemma 13. *For any* $n \geq 32$ *and* $\rho \in [1/n, 1/32]$*, there is a graph G on* $n+1$ *vertices such that any spanner H of G with lightness at most* $1+\rho$ *has average distortion at least* $\Omega(1/\rho)$.

Proof. We define the graph $G = (V, E)$ as follows. Denote $V = \{v_0, v_1, \ldots, v_n\}$, $E = \binom{V}{2}$, and the weight function w is defined as follows.

$$w(\{v_i, v_j\}) = \begin{cases} 1 & \text{if } |i-j| = 1 \\ 2 & \text{otherwise .} \end{cases}$$

I.e., G is a complete graph of size $n+1$, where the edges $\{v_i, v_{i+1}\}$ have unit weight and induce a path of length n, and all non-path edges have weight 2. Clearly, the path is the MST of G of weight n. Let $k = \lceil \rho n \rceil$. Let H be some spanner of G with lightness at most $1 + \rho \leq \frac{n+k}{n}$, in particular, $w(H) \leq n + k$. Clearly H has at least n edges (to be connected). Let q be the number of edges of weight 2 contained in H. Then $w(H) \geq (n-q) \cdot 1 + q \cdot 2 = n + q$. Therefore $q \leq k$.

Let S be the set of vertices which are incident on an edge of weight 2 in H. Then $|S| \leq 2q \leq 2k$. Let $\delta = \frac{1}{32\rho}$. For any $v \in S$, let $N_v \subseteq V$ be the set of vertices that are connected to v via a path of length at most δ in H, such that this path consists of weight 1 edges only. Necessarily, for any $v \in S$, $|N_v| \leq 2\delta + 1$. Let $N = \bigcup_{v \in S} N_v$, it holds that $|N| \leq 2k \cdot (2\delta + 1) \leq 4\rho n(\frac{1}{16\rho} + 1) \leq \frac{n}{4} + \frac{n}{8} = \frac{3n}{8}$. Let $\bar{N} = V \setminus N$.

Consider $u \in \bar{N}$. By definition of N every weight 2 edge is further than δ steps away from u in H. It follows that there are at most $2\delta + 1$ vertices within distance at most δ from u (in H). Let $F_u = \{v \in V \ : \ d_H(u, v) > \delta\}$. It follows that $|F_u| \geq n - 2\delta - 1$. Note that for any $v \in F_u$, the distortion of the pair $\{u, v\}$ is at least $\frac{\delta}{2}$. Hence, we obtain that

$$\begin{aligned} \sum_{\{v,u\} \in \binom{V}{2}} \frac{d_H(v,u)}{d_G(v,u)} &\geq \frac{1}{2} \sum_{u \in \bar{N}} \sum_{v \in F_u} \frac{d_H(v,u)}{d_G(v,u)} \\ &\geq \frac{5n}{16} \cdot (n - 2\delta - 1) \cdot \frac{\delta}{2} \\ &\geq \frac{5n}{16} \cdot \frac{7n}{8} \cdot \frac{1}{64\rho} . \end{aligned}$$

Finally, the average distortion is bounded as follows.

$$\text{dist}_1(H) = \frac{1}{\binom{n+1}{2}} \sum_{\{v,u\} \in \binom{V}{2}} \frac{d_H(v,u)}{d_G(v,u)}$$

[17] The fact the embedding yields coarse scaling distortion is implicit in their proof.

[18] We also mention that in general the average distortion of a spanner cannot be arbitrarily close to 1, unless the spanner is extremely dense. E.g., when G is a complete graph, any spanner with lightness at most $n/4$ will have average distortion at least $3/2$.

$$\geq \frac{n}{n+1} \cdot \frac{35}{64} \cdot \frac{1}{64\rho}$$
$$\geq \frac{1}{128\rho}. \quad \square$$

7. Conclusions

In this paper we constructed a spanning tree with $1+\rho$ lightness and $O(\frac{1}{\rho})$ average distortion for any parameter $\rho \in (0,1)$. We also proved that this tradeoff is best possible, up to constant factors. The main technical contribution is a new equivalence theorem between prioritized and coarse scaling distortions. We used this equivalence theorem in order to derive several other results in metric embeddings. In particular, we show that every finite metric space embeds into ℓ_p space with prioritized distortion $O(\log j)$. We hope that this equivalence theorem will lead to additional results and applications.

Acknowledgment

We are grateful to Michael Elkin and Shiri Chechik for fruitful discussions.

References

[1] Ittai Abraham, Yair Bartal, Hubert T.-H. Chan, Kedar Dhamdhere, Anupam Gupta, Jon M. Kleinberg, Ofer Neiman, Aleksandrs Slivkins, Metric embeddings with relaxed guarantees, in: 46th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2005, 23-25 October 2005, Pittsburgh, PA, USA, Proceedings, 2005, pp. 83–100.
[2] Ittai Abraham, Yair Bartal, Ofer Neiman, Nearly tight low stretch spanning trees, in: Proceedings of the 49th Annual IEEE Symposium on Foundations of Computer Science, FOCS '08, IEEE Computer Society, Washington, DC, USA, 2008, pp. 781–790.
[3] Ittai Abraham, Yair Bartal, Ofer Neiman, Advances in metric embedding theory, Adv. Math. 228 (6) (2011) 3026–3126.
[4] Ittai Abraham, Yair Bartal, Ofer Neiman, Embedding metrics into ultrametrics and graphs into spanning trees with constant average distortion, SIAM J. Comput. 44 (1) (2015) 160–192.
[5] Ittai Abraham, Yair Bartal, Ofer Neiman, Leonard J. Schulman, Volume in general metric spaces, Discrete Comput. Geom. 52 (2) (2014) 366–389.
[6] B. Awerbuch, A. Baratz, D. Peleg, Efficient Broadcast and Light-Weight Spanners, Technical Report CS92-22, The Weizmann Institute of Science, Rehovot, Israel, 1992.
[7] Stephen Alstrup, Søren Dahlgaard, Arnold Filtser, Morten Stöckel, Christian Wulff-Nilsen, Constructing light spanners deterministically in near-linear time, CoRR, arXiv:1709.01960, 2017.
[8] Noga Alon, Richard M. Karp, David Peleg, Douglas West, A graph-theoretic game and its application to the k-server problem, SIAM J. Comput. 24 (1) (1995) 78–100.
[9] Ittai Abraham, Ofer Neiman, Using petal-decompositions to build a low stretch spanning tree, in: Proceedings of the Forty-fourth Annual ACM Symposium on Theory of Computing, STOC '12, ACM, New York, NY, USA, 2012, pp. 395–406.
[10] Yair Bartal, Probabilistic approximation of metric spaces and its algorithmic applications, in: Proceedings of the 37th Annual Symposium on Foundations of Computer Science, IEEE Computer Society, Washington, DC, USA, 1996, p. 184.
[11] Yair Bartal, On approximating arbitrary metrices by tree metrics, in: Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing, STOC '98, ACM, New York, NY, USA, 1998, pp. 161–168.
[12] Yair Bartal, Graph decomposition lemmas and their role in metric embedding methods, in: Algorithms - ESA 2004, 12th Annual European Symposium, Bergen, Norway, September 14-17, 2004, Proceedings, 2004, pp. 89–97.
[13] Yair Bartal, Arnold Filtser, Ofer Neiman, On notions of distortion and an almost minimum spanning tree with constant average distortion, in: Proceedings of the Twenty-seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA '16, Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2016, pp. 873–882.
[14] Yair Bartal, Nova Fandina, Ofer Neiman, On moment analysis of metric embedding and its application in dimensionality reduction, 2017, manuscript.
[15] Jean Bourgain, On Lipschitz embedding of finite metric spaces in Hilbert space, Isr. J. Math. 52 (1–2) (1985) 46–52.
[16] T.-H. Hubert Chan, Michael Dinitz, Anupam Gupta, Spanners with slack, in: Proceedings of the 14th Conference on Annual European Symposium - Volume 14, ESA'06, Springer-Verlag, London, UK, 2006, pp. 196–207.
[17] Barun Chandra, Gautam Das, Giri Narasimhan, José Soares, New sparseness results on graph spanners, in: Proceedings of the Eighth Annual Symposium on Computational Geometry, SCG '92, ACM, New York, NY, USA, 1992, pp. 192–201.
[18] Shiri Chechik, Approximate distance oracles with improved bounds, in: Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing, STOC 2015, Portland, OR, USA, June 14-17, 2015, 2015, pp. 1–10.
[19] Shiri Chechik, Christian Wulff-Nilsen, Near-optimal light spanners, ACM Trans. Algorithms 14 (3) (2018) 33:1–33:15.
[20] Michael Elkin, Yuval Emek, Daniel A. Spielman, Shang-Hua Teng, Lower-stretch spanning trees, SIAM J. Comput. 38 (2) (2008) 608–628.
[21] Michael Elkin, Arnold Filtser, Ofer Neiman, Terminal embeddings, Theor. Comput. Sci. 697 (2017) 1–36.
[22] Michael Elkin, Arnold Filtser, Ofer Neiman, Prioritized metric structures and embedding, SIAM J. Comput. 47 (3) (2018) 829–858.
[23] Michael Elkin, Ofer Neiman, Shay Solomon, Light spanners, SIAM J. Discrete Math. 29 (3) (2015) 1312–1321.
[24] Jittat Fakcharoenphol, Satish Rao, Kunal Talwar, A tight bound on approximating arbitrary metrics by tree metrics, J. Comput. Syst. Sci. 69 (3) (2004) 485–497.
[25] Arnold Filtser, Shay Solomon, The greedy spanner is existentially optimal, in: Proceedings of the 2016 ACM Symposium on Principles of Distributed Computing, PODC 2016, Chicago, IL, USA, July 25–28, 2016, 2016, pp. 9–17.
[26] E.N. Gilbert, H.O. Pollak, Steiner minimal trees, SIAM J. Appl. Math. 16 (1) (Jan 1968) 1–29.
[27] Samir Khuller, Balaji Raghavachari, Neal E. Young, Balancing minimum spanning and shortest path trees, in: SODA, 1993, pp. 243–250.
[28] Jon Kleinberg, Aleksandrs Slivkins, Tom Wexler, Triangulation and embedding using small sets of beacons, J. ACM 56 (6) (September 2009) 32:1–32:37.

Part IV

Steiner Point Removal with distortion $O(\log k)$, using the Relaxed-Voronoi algorithm

SIAM J. COMPUT.
Vol. 48, No. 2, pp. 249–278

STEINER POINT REMOVAL WITH DISTORTION $O(\log k)$ USING THE RELAXED-VORONOI ALGORITHM*

ARNOLD FILTSER†

Abstract. In the Steiner point removal problem, we are given a weighted graph $G = (V, E)$ and a set of terminals $K \subset V$ of size k. The objective is to find a minor M of G with only the terminals as its vertex set, such that distances between the terminals will be preserved up to a small multiplicative distortion. Kamma, Krauthgamer, and Nguyen [*SIAM J. Comput.*, 44 (2015), pp. 975–995] devised a ball-growing algorithm with exponential distributions to show that the distortion is at most $O(\log^5 k)$. Cheung [*Proceedings of the* 29*th Annual ACM/SIAM Symposium on Discrete Algorithms*, 2018, pp. 1353–1360] improved the analysis of the same algorithm, bounding the distortion by $O(\log^2 k)$. We devise a novel and simpler algorithm (called the `Relaxed-Voronoi` algorithm) which incurs distortion $O(\log k)$. This algorithm can be implemented in almost linear time ($O(|E| \log |V|)$).

Key words. Steiner point removal (SPR), distortion, metric embedding, minor graph, randomized algorithm

AMS subject classifications. 41, 60, 68

DOI. 10.1137/18M1184400

1. Introduction.

In graph compression problems the input is usually a massive graph. The objective is to compress the graph into a smaller graph, while preserving certain properties of the original graph, such as distances or cut values. Compression allows us to obtain faster algorithms while reducing the storage space. In the era of massive data, the benefits are obvious. Examples of such structures are graph spanners [37], distance oracles [39], cut sparsifiers [7], spectral sparsifiers [6], and vertex sparsifiers [36].

In this paper we study the *Steiner point removal* (SPR) problem. Here we are given an undirected graph $G = (V, E)$ with positive weight function $w : E \to \mathbb{R}_+$, and a subset of terminals $K \subseteq V$ of size k (the nonterminal vertices are called Steiner vertices). The goal is to construct a new graph $M = (K, E')$ with positive weight function w', with the terminals as its vertex set, such that (1) M is a graph minor of G and (2) the distance between every pair of terminals t, t' is distorted by at most a multiplicative factor of α, formally

$$\forall t, t' \in K, \quad d_G(t, t') \le d_M(t, t') \le \alpha \cdot d_G(t, t') \ .$$

Property (1) expresses preservation of the topological structure of the original graph. For example, if G was planar, so will M be. Property (2), however, expresses preservation of the geometric structure of the original graph, that is, distances between terminals. The question is, What is the minimal α (which may depend on k) such that every graph with a terminal set of size k will admit a solution to the SPR problem with distortion α?

The first to study a problem of this flavor was Gupta [24], who showed that given a weighted tree T with a subset of terminals K, there is a tree T' with K as its vertex

*Received by the editors April 30, 2018; accepted for publication January 22, 2019; published electronically March 26, 2019. A preliminary version appeared in *Proceedings of SODA*'18, 2018.
http://www.siam.org/journals/sicomp/48-2/M118440.html
Funding: The research was supported in part by ISF grant (1718/18) and BSF grant 2015813.
†Department of Computer Science, Ben Gurion University of the Negev, Beer Sheva, 8410501, Israel (arnoldf@cs.bgu.ac.il).

set that preserves all the distances between terminals up to a multiplicative factor of 8. Chan et al. [9] observed that the tree T' of Gupta is in fact a minor of the original tree T. They showed that 8 is the best possible distortion and formulated the problem for general graphs. This lower bound of 8 is achieved on the complete unweighted binary tree and is the best known lower bound for the general SPR problem.

Basu and Gupta [5] showed that on outerplanar graphs, the SPR problem can be solved with distortion $O(1)$.

Kamma, Krauthgamer, and Nguyen were the first to bound the distortion for general graphs. They suggested the `Ball-growing` algorithm. Their first analysis provide $O(\log^6 k)$ distortion (conference version [26]), which they later improved to $O(\log^5 k)$ (journal version [27]). Recently, Cheung [11] improved the analysis of the `Ball-growing` algorithm further, providing an $O(\log^2 k)$ upper bound on the distortion.

The `Ball-growing` algorithm constructs a terminal partition, that is, a partition where each cluster is connected and contains a single terminal. The minor is then constructed by contracting all the internal edges in all clusters. The weight of the minor edge $\{t, t'\}$ (if it exists) is defined simply to $d_G(t, t')$. The clusters are generated iteratively. In each round, by turn, each terminal t_j increases the radius R_j of its ball cluster V_j in an attempt to add more vertices to its ball cluster V_j. Once a vertex joins some cluster, it will remain there. In round ℓ, the radii are (independently) distributed according to an exponential distribution, where the mean of the distribution grows in each round. A description of the `Ball-growing` algorithm can be found in Appendix B.

The main contribution of this paper is a new upper bound of $O(\log k)$ for the SPR problem. In a preliminary conference version [20], the author improved the analysis of the `Ball-growing` algorithm, providing an $O(\log k)$ upper bound. In this paper we devise a novel algorithm called the `Relaxed-Voronoi` algorithm. We bound the distortion incurred by the minor produced using the `Relaxed-Voronoi` by $O(\log k)$ as well. Nevertheless, the `Relaxed-Voronoi` algorithm is arguably simpler and more intuitive compared to the `Ball-growing` algorithm. Both algorithms grow clusters around the terminals; the main difference is that the `Ball-growing` algorithm has many iterations, growing slowly from all terminals (almost in parallel), while the `Relaxed-Voronoi` algorithm has one round only (the terminals create clusters by turns. Once a cluster is created it will remain unchanged till the end of the algorithm). The analysis in [20] was built upon [11]. In both papers, a considerable effort was made to lower and upper bound the number of the round in which each nonterminal is clustered. The analysis in this paper is quite similar to [20], while all the round-base analysis simply becomes unnecessary.

Furthermore, we devise an efficient implementation of the `Relaxed-Voronoi` algorithm in almost linear time $O(m + \min\{m, nk\} \cdot \log n)$ (m (resp., n) here is the number of edges (resp., vertices) in G). While the `Ball-growing` algorithm can be implemented in polynomial time, it is not clear how to do so efficiently.

We show that the analysis of the `Relaxed-Voronoi` algorithm is asymptotically tight. That is, there are graphs for which the `Relaxed-Voronoi` produces a minor which incurs distortion $\Omega(\log k)$. We prove a similar lower bound also for the `Ball-growing` algorithm. However, there we are only able to prove an $\Omega(\sqrt{\log k})$ lower bound on the performance of the algorithm.

1.1. Related work. Englert et al. [17] showed that every graph G admits a distribution $\mathcal{D}$ over terminal minors with expected distortion $O(\log k)$. Formally, for all $t_i, t_j \in K$, it holds that $1 \le \frac{\mathbb{E}_{M\sim\mathcal{D}}[d_M(t_i,t_j)]}{d_G(t_i,t_j)} \le O(\log k)$. Thus, Theorem 3.1 can be seen as an improvement upon [17], where we replace distribution with a single minor. Englert et al. showed better results for β-decomposable graphs; in particular, they showed that graphs excluding a fixed minor admit a distribution with $O(1)$ expected distortion.

Krauthgamer, Nguyen, and Zondiner [29] showed that if we allow the minor M to contain at most $\binom{k}{2}^2$ Steiner vertices (in addition to the terminals), then distortion 1 can be achieved. They further showed that for graphs with constant treewidth, $O(k^2)$ Steiner points will suffice for distortion 1. Cheung, Goranci, and Henzinger [12] showed that allowing $O(k^{2+\frac{2}{t}})$ Steiner vertices, one can achieve distortion $2t-1$ (in particular distortion $O(\log k)$ with $O(k^2)$ Steiners). For planar graphs, they achieved $1+\epsilon$ distortion with $\tilde{O}((\frac{k}{\epsilon})^2)$ Steiner points.

There is a long line of work focusing on preserving the cut/flow structure among the terminals by a graph minor. See [36, 32, 10, 34, 17, 13, 30, 2, 23, 31].

There are works studying metric embeddings and metric data structures concerning preserving distances among terminals, or from terminals to other vertices, out of the context of minors. See [14, 38, 25, 28, 15, 16, 4].

Finally, there are clustering algorithms similar in nature to the `Relaxed-Voronoi` and `Ball-growing` algorithms [33, 3, 19, 8, 18, 35].

1.2. Technical ideas. The basic approach in this paper, as well as in all previous papers on SPR in general graphs, is to use terminal partitions in order to construct a minor for the SPR problem. Specifically, we partition the vertices into k connected clusters, with a single terminal in each cluster. Such a partition induces a minor by contracting all the internal edges in each cluster. See the preliminaries for more details. Considering such a framework, the most natural idea will be to partition the vertices into the Voronoi cells, i.e., the cluster V_j of the terminal t_j will contain all the vertices v for which t_j is the closest terminal. However, this approach miserably fails and can incur distortion as large as $k-1$. See Figure 1.1 for an illustration.

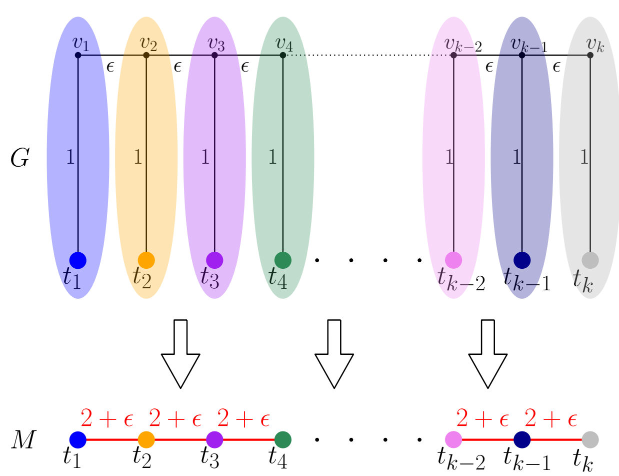

FIG. 1.1. *The graph G consists of a k-path of Steiner vertices $v_1,\ldots,v_k$ with edges of weight ϵ. To each Steiner vertex v_j we add a terminal using a unit weight edge. The Voronoi cell of the terminal t_j is $\{t_j,v_j\}$. The minor M induced by this terminal partition is a path $t_1,\ldots,t_k$ where the weight of each edge equals $2+\epsilon$. The original distance in G between t_1 to t_k is $2+(k-1)\cdot\epsilon$, while the distance in the minor M equals $(k-1)\cdot(2+\epsilon)$. In particular, when ϵ tends to 0, the distortion tends to $k-1$.*

Our idea is to introduce some noise in order to avoid the sharp boundaries between the clusters. Specifically, we order the terminals in an arbitrary order. For each terminal t_j we sample a parameter $R_j \geq 1$ that we will call its *magnitude*. Then, by turn, each terminal will construct a cluster V_j which will be essentially a magnified (by R_j) Voronoi cell (in the remaining graph). However, in order to maintain connectivity, the magnified Voronoi cell is constructed in a "Dijkstra manner" as follows. For every vertex v, denote by $D(v)$ the distance from v to its closest terminal. Initially $V_j = \{t_j\}$. In each step, every unclustered neighboring vertex v of V_j is examined. If $d_G(v, t_j) \leq R_j \cdot D(v)$, then v joins the cluster V_j. The process terminates when no new potential vertices remain. Then we move on to the next terminal and repeat the same process on the remaining graph. Eventually, all of G is partitioned into clusters.

To sample R_j, we first sample g_j according to geometric distribution with parameter $p = \frac{1}{5}$. Then, R_j is set to be $(1+\delta)^{g_j}$, where $\delta = \Theta(\frac{1}{\ln k})$. In particular, all the R_j's are bounded by some universal constant with high probability (w.h.p.).

Next, we provide some intuition for the distortion analysis. Consider a pair of terminals t, t', and let $P_{t,t'}$ be the shortest path between them in the original graph G. When the algorithm terminates, all the vertices in $P_{t,t'}$ are clustered by different terminals. See Figure 4.2 for an illustration. Let $\mathcal{D}_{\ell_1}, \dots, \mathcal{D}_{\ell_k}$ be the partition of the vertices in $P_{t,t'}$ induced by the partition of all vertices created by the algorithm. i.e., $\mathcal{D}_{\ell_i} = P_{t,t'} \cap V_{\ell_i}$. For simplicity at this stage, we will assume that every $\mathcal{D}_{\ell_j}$ is continuous. In the induced minor graph, there is an edge between any two consecutive terminals t_{ℓ_j} and $t_{\ell_{j+1}}$. Therefore the distance between t and t' in the minor graph can be bounded by $\sum_j d_G(t_{\ell_j}, t_{\ell_{j+1}})$. Let v^{ℓ_j} be the "first" vertex on $P_{t,t'}$ to be covered by t_{ℓ_j}. "First" here is in the following sense: we think about the sampling of R_j in a gradual manner. For a vertex v, let r_v denote the minimal value of R_j such that $v \in V_j$. Then v^j is defined to be the vertex with the minimal value r_v. Using the triangle inequality, $d_G(t_{\ell_j}, t_{\ell_{j+1}}) \leq d_G(t_{\ell_j}, v^{\ell_j}) + d_G(v^{\ell_j}, v^{\ell_{j+1}}) + d_G(v^{\ell_{j+1}}, t_{\ell_{j+1}})$. Therefore $d_M(t, t') \leq \sum_{i=1}^{k'-1} d_G(v^{\ell_i}, v^{\ell_{i+1}}) + 2\sum_{i=1}^{k'} d_G(t_{\ell_i}, v^{\ell_i}) \leq d_G(t, t') + 2\sum_{i=1}^{k'} d_G(t_{\ell_i}, v^{\ell_i})$ (see Figure 4.2 for an illustration).

In order to bound the distortion, we need to bound the sum of "deviations" $\sum_{i=1}^{k'} d_G(t_{\ell_i}, v^{\ell_i})$ from the shortest path. However, these deviations are heavily dependent. Instead of analyzing the deviations directly, we will follow an approach first suggested by [11]. We partition the shortest path $P_{t,t'}$ from t to t' into a set of intervals $\mathcal{Q}$; the idea will be to count for each interval Q how many deviations start from this interval (denoted $X(Q)$). Specifically, for each deviation, we will charge the interval in which this deviation was initiated. Afterward, we will be able to replace the sum of deviations above by a linear combination of the interval charges.

The partition of the shortest path $P_{t,t'}$ into intervals is done such that the length of each interval $Q \in \mathcal{Q}$ will be a $\log k$ fraction of the distance from the interval to its closest terminal. Such interval lengths will ensure the following crucial property: given that some vertex $v \in Q$ joins the cluster V_j (of the terminal t_j), with probability at least $1 - p$, all of Q joins V_j.

Using this property alone, one can show that the expected charge on each interval is bounded by a constant. This already will imply an $O(\log k)$ distortion on each pair in expectation. However, as we are interested in $O(\log k)$ distortion on all pairs w.h.p., a more subtle argument is required. We couple the interval charges into a series of independent random variables that dominate the interval charges. Then, a concentration bound on the independent variables implies an upper bound on the sum of interval charges, which provides $O(\log k)$ distortion w.h.p.

1.3. Paper organization. In section 3 we describe the `Relaxed-Voronoi` algorithm and prove some of its basic properties. Then, in section 4 we analyze the distortion incurred by the `Relaxed-Voronoi` algorithm. In section 5 we introduce a small modification to the `Relaxed-Voronoi` algorithm. We prove that the distortion analysis is still valid and explain how the modified algorithm can be efficiently implemented. In section 6 we prove that our analysis of the `Relaxed-Voronoi` algorithm is asymptotically tight (and provide some lower bound on the performance of the `Ball-growing` algorithm). Finally, in section 7 we provide some concluding remarks and discuss further directions.

2. Preliminaries. Appendix C contains a summary of all the definitions and notation we use. The reader is encouraged to refer to this index while reading.

We consider undirected graphs $G = (V, E)$ with positive edge weights $w : E \to \mathbb{R}_{\geq 0}$. Let d_G denote the shortest path metric in G. For a subset of vertices $A \subseteq V$, let $G[A]$ denote the *induced graph* on A. Fix $K = \{t_1, \dots, t_k\} \subseteq V$ to be a set of *terminals*. For a vertex v, $D(v) = \min_{t \in K} d_G(v, t)$ is the distance from v to its closest terminal. For clarity, we will assume that all metric distances are unique (that is, for $\{v, v'\} \neq \{u, u'\}$, $d_G(v, v') \neq d_G(u, u')$). Moreover, we will assume that for every pair v, u there is a unique shortest path. Otherwise, we can introduce arbitrarily small perturbations.

A graph H is a *minor* of a graph G if we can obtain H from G by edge deletions/contractions and vertex deletions. A partition $\{V_1, \dots, V_k\}$ of V is called a *terminal partition* (w.r.t. K) if for every $1 \leq i \leq k$, $t_i \in V_i$, and the induced graph $G[V_i]$ is connected. See Figure 2.1 for an illustration. The *induced minor* by terminal partition $\{V_1, \dots, V_k\}$ is a minor M, where each set V_i is contracted into a single vertex called (abusing notation) t_i. Note that there is an edge in M from t_i to t_j iff there are vertices $v_i \in V_i$ and $v_j \in V_j$ such that $\{v_i, v_j\} \in E$. We determine the weight of the edge $\{t_i, t_j\} \in E(M)$ to be $d_G(t_i, t_j)$. Note that by the triangle inequality, for every pair of (not necessarily neighboring) terminals t_i, t_j, it holds that $d_M(t_i, t_j) \geq d_G(t_i, t_j)$. The *distortion* of the induced minor is $\max_{i,j} \frac{d_M(t_i,t_j)}{d_G(t_i,t_j)}$.

2.1. Probability. For a distribution $\mathcal{D}$, $X \sim \mathcal{D}$ denotes that X is a random variable distributed according to $\mathcal{D}$.

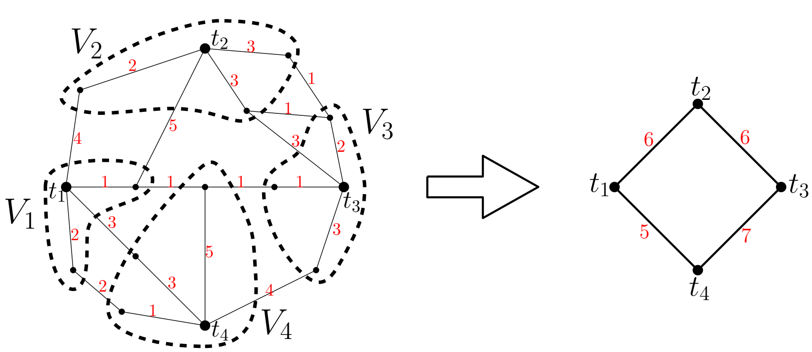

FIG. 2.1. *The left side of the figure contains a weighted graph $G = (V, E)$, with weights specified in red, and four terminals $\{t_1, t_2, t_3, t_4\}$. The dashed black curves represent a terminal partition of the vertex set V into the subsets V_1, V_2, V_3, V_4. The right side of the figure represent the minor M induced by the terminal partition. The distortion is realized between t_1 and t_3, and is* $\frac{d_M(t_1,t_3)}{d_G(t_1,t_3)} = \frac{12}{4} = 3$.

$\mathsf{Geo}(p)$ denotes the *geometric distribution* with parameter p. Here we toss a biased coin with probability p for heads, until the first time we get heads. $\mathsf{Geo}(p)$ is the number of coin tosses. Formally, $\mathsf{Geo}(p)$ is supported in $\{1, 2, 3, \dots\}$, where the probability to get s is $(1-p)^{s-1} \cdot p$.

Exponential distribution is the continuous analogue of geometric distribution. $\mathsf{Exp}(\lambda)$ denotes the exponential distribution with mean λ and density function $f(x) = \frac{1}{\lambda} e^{-\frac{x}{\lambda}}$ for $x \geq 0$. Exponential distribution is closed under scaling, that is, for $X \sim \mathsf{Exp}(\lambda)$, $c \cdot X$ is distributed according to $\mathsf{Exp}(c\lambda)$. We will use the following concentration bound.

LEMMA 2.1. *Suppose $X_1, \dots, X_n$'s are independent random variables, where each X_i is distributed according to $\mathsf{Exp}(\lambda_i)$. Let $X = \sum_i X_i$ and $\lambda_M = \max_i \lambda_i$. Set $\mu = \mathbb{E}[X] = \sum_i \lambda_i$.*

$$\text{For } a \geq 2\mu, \qquad \Pr[X \geq a] \leq \exp\left(-\frac{1}{2\lambda_M}(a - 2\mu)\right) .$$

In Appendix A we prove a more general bound. In particular, Lemma 2.1 above is a special case of Lemma A.1 (which is obtained by choosing parameters $\alpha = \frac{a}{\mu} - 1$ and $t = \frac{1}{2\lambda_M}$).

3. Algorithm. The terminals are ordered in arbitrary order $t_1, t_2, \dots, t_k$. The `Relaxed-Voronoi` algorithm has k rounds, where in the round i, the cluster V_i (containing t_i) is constructed in the graph induced by the non-terminal vertices not clustered so far.

The clusters are created using the `Create-Cluster` procedure. The algorithm provides a random variable $R_j = (1+\delta)^{g_j}$, where g_j is distributed according to geometric distribution with parameter p.

The `Create-Cluster` procedure runs in a Dijkstra-like fashion. During the execution, we maintain three sets: (1) V_j: the currently created cluster (initiated to be $\{t_j\}$). (2) U: the set of vertices that were "refused" to join V_j. (3) N: the set of neighboring vertices to V_j (that are not in U).

While N is nonempty, the algorithm extracts an arbitrary vertex v from N. If $d_G(v, t_j) \leq R(j) \cdot D(v)$ (the distance from t_j to v is at most R_j times the distance from v to its closest terminal), then v joins V_j. Otherwise v joins U. In the case where v joins V_j, all its neighbors (outside of $U \cup V_j$) join N. As each vertex might join N at most once, eventually N becomes empty. Then the procedure ceases and returns V_j.

THEOREM 3.1. *With probability $1 - \frac{1}{k}$, in the minor graph M returned by Algorithm* 3.1*, it holds that for every two terminals t, t', $d_M(t, t') \leq O(\log k) \cdot d_G(t, t')$.*

First we argue that Algorithm 3.1 indeed produces a terminal partition.

LEMMA 3.2. *The sets $V_1, \dots, V_k$ constructed by Algorithm* 3.1 *form a terminal partition.*

Proof. It is straightforward from the description of the algorithm that the sets $V_1, \dots, V_k$ are disjoint and that for every j, $t_j \in V_j$ and $G[V_j]$ is connected. The only nontrivial property we have to show is that every vertex $v \in V$ joins some cluster.

Fix some $v \in V$, let t_j be the closest terminal to v (s.t. $D(v) = d_G(v, t_j)$), and let $P = \{t_j = u_0, u_1, \dots, u_s = v\}$ be the shortest path from t_j to v in G. Note that as P is a shortest path, t_j is also the closest terminal to all the vertices in P. As $t_j = u_0 \in V_j$, at least one vertex from P is clustered during the algorithm. Let $u_{i'}$ be

Algorithm 3.1. $M = \texttt{Relaxed-Voronoi}(G = (V, E, w), K = \{t_1, \ldots, t_k\})$.

```
1: Set δ = 1/(20 ln k) and p = 1/5.
2: Set V⊥ ← V \ K.                    // V⊥ is the currently unclustered vertices.
3: for j from 1 to k do
4:    Choose independently at random g_j distributed according to Geo(p).
5:    Set R_j ← (1 + δ)^{g_j}.
6:    Set V_j ← Create-Cluster(G, V⊥, t_j, R_j).
7:    Remove all the vertices in V_j from V⊥.
8: end for
9: return the terminal-centered minor M of G induced by V_1, ..., V_k.
```

the first clustered vertex from P (w.r.t. time). Denote by $V_{j'}$ the cluster $u_{i'}$ joins to. We argue by induction on $i \geq i'$ that u_i also joins $V_{j'}$. This will imply that $u_s = v$ joins $V_{j'}$ and thus is clustered. Suppose u_i joins $V_{j'}$. It holds that $d_G(u_i, t_{j'}) \leq R_{j'} \cdot D(u_i)$. Moreover, all the neighbors of u_i join N. Therefore u_{i+1} necessarily joined to the set N (at some stage during the execution of the `Create-Cluster` procedure for $V_{j'}$). As

$$\begin{aligned} d_G(u_{i+1}, t_{j'}) &\leq d_G(u_{i+1}, u_i) + d_G(u_i, t_{j'}) \\ &\leq d_G(u_{i+1}, u_i) + R_{j'} \cdot d_G(u_i, t_j) \\ &\leq R_{j'} \cdot d_G(u_{i+1}, t_j) = R_{j'} \cdot D(u_{i+1}) \ , \end{aligned}$$

u_{i+1} will join $V_{j'}$, as required. ∎

3.1. Modification. Let $\hat{\Delta} = \min_{t,t' \in K}\{d_G(t, t')\}$ denote the minimal distance between a pair of terminals. Note that $\hat{\Delta} > 0$. For the sake of analysis we will make a preprocessing step to ensure that every edge e has weight at most $c_w \cdot \hat{\Delta} = \frac{\delta}{24} \cdot \hat{\Delta}$. This can be achieved by subdividing larger edges, i.e., adding additional vertices of degree two in the middle of such edges. Denote by $\hat{G}$ the modified graph G, when we repeatedly subdivide edges until every edge e has small enough weight. We argue that such subdivisions did not affect whatsoever the terminal-centered minor returned by Algorithm 3.1.

CLAIM 3.3. *Let $G = (V, E, w)$ be a weighted graph with terminal set $K = \{t_1, \ldots, t_k\}$. Consider an edge $e = \{v, u\} \in E$ of weight ω. Let $\tilde{G}$ be the graph G with subdivided edge e. Specifically, we add a new Steiner vertex v_e and replace the edge e by two new edges $\{v_e, v\}$, $\{v_e, u\}$, both of weight $\omega/2$.*

Fix $g_1, \ldots, g_k$ and consider Algorithm 3.1*, where the random choices in line* 4 *are $g_1, \ldots, g_k$, respectively. Then the terminal-centered minor M returned on input G is the same as the terminal-centered minor $\tilde{M}$ returned on input $\tilde{G}$.*

Proof. As $g_1, \ldots, g_k$ are fixed, Algorithm 3.1 is now deterministic. Let $V_1, \ldots, V_k$ be the terminal partition induced by Algorithm 3.1 on G, and similarly let $\tilde{V}_1, \ldots, \tilde{V}_k$ be the terminal partition induced by Algorithm 3.1 on $\tilde{G}$. We argue that for all j, $V_j = \tilde{V}_j \setminus \{v_e\}$. Note that this will imply our claim. Indeed, let V_j, $V_{j'}$ be the clusters such that $v \in V_j$ and $u \in V_{j'}$. As each cluster is connected, necessarily $v_e \in V_j \cup V_{j'}$. By the definition of subdivision, this will imply that the terminal-centered minors are indeed identical.

Each Steiner vertex can be clustered only after at least one of its neighbors is clustered. Therefore v_e cannot be clustered before both v and u. Without loss of generality (w.l.o.g.) v joined V_j while u is still unclustered. The vertex v_e wasn't

Algorithm 3.2. $V_j = \texttt{Create-Cluster}(G = (V, E, w), V_{\perp}, t_j, R_j)$.

1: Set $V_j \leftarrow \{t_j\}$.
2: Set $U \leftarrow \emptyset$. // *U is the set of vertices already denied from V_j.*
3: Set N to be all the neighbors of t_j in $V_{\perp}$.
4: **while** $N \neq \emptyset$ **do**
5: Let v be an arbitrary vertex from N.
6: Remove v from N.
7: **if** $d_G(v, t_j) \leq R_j \cdot D(v)$ **then**
8: Add v to V_j.
9: Add all the neighbors of v in $V_{\perp} \setminus (U \cup V_j)$ to N.
10: **else**
11: Add v to U.
12: **end if**
13: **end while**
14: **return** V_j.

examined before the clustering of v. Denote by V'_j (resp., $\tilde{V}'_j$) the set V_j (resp., $\tilde{V}_j$) right after the clustering of v at the execution of Algorithm 3.1 on G (resp., $\tilde{G}$). Note that the order of extraction from N in line 5 of Algorithm 3.2 is determined deterministically. Therefore, up to the clustering of v the algorithm behaved the same on both G and $\tilde{G}$. In particular, for all $j'' < j$, $V_{j''} = \tilde{V}_{j''}$. Moreover, $V'_j = \tilde{V}'_j$. After v joins V_j, v_e joins (for the first time) to the set N (for $\tilde{G}$). Note that

$$D(v_e) = \min\{D(v), D(u)\} + \frac{\omega}{2},$$
$$d_G(t_j, v_e) = \min\{d_G(t_j, v), d_G(t_j, u)\} + \frac{\omega}{2}.$$

As v joined V_j, necessarily $d_G(t_j, v) \leq R_j \cdot D(v)$. Consider the following cases:

- $u \notin V_j$: In the algorithm for G, u was examined (as $v \in V_j$), thus $d_G(t_j, u) > R_j \cdot D(u)$. Therefore u will also not join $\tilde{V}_j$. As v_e has edges only to v and u, v_e has no impact on any other vertex. Therefore the cluster $\tilde{V}_j$ will be constructed in the same manner as V_j (up to maybe containing v_e). Note that all the other clusters will not be affected, as if v_e remained unclustered, it becomes a leaf. We conclude that for every j'', $V_{j''} = \tilde{V}_{j''} \setminus \{v_e\}$.
- $u \in V_j$: It holds that $d_G(t_j, u) \leq R_j \cdot D(u)$. Therefore

$$d_G(t_j, v_e) = \min\{d_G(t_j, v), d_G(t_j, e)\} + \frac{\omega}{2} \leq R_j \cdot \min\{D(v), D(u)\} + \frac{\omega}{2} \leq R_j \cdot D(v_e)\ .$$

Therefore v_e will join $\tilde{V}_j$, which will ensure that u joins $\tilde{N}$, and afterward to $\tilde{V}_j$. Note that v_e has no other impact. In particular, for every $j'' \neq j$, $V_{j''} = \tilde{V}_{j''}$ while $V_j \cup \{v_e\} = \tilde{V}_j$. $\square$

Consider the modified graph $\hat{G}$. Suppose that we proved that with probability at least $1 - \frac{1}{k}$, in the minor graph $\hat{M}$ returned by Algorithm 3.1 for $\hat{G}$, it holds that for every two terminals t, t', $d_{\hat{M}}(t, t') \leq O(\log k) \cdot d_{\hat{G}}(t, t') = O(\log k) \cdot d_G(t, t')$. Then by repetitive use of Claim 3.3 (once for every new vertex), Theorem 3.1 follows. From now on, we will abuse notation and refer to the graph $\hat{G}$ as G. Note that all this is done purely for the sake of analysis, as by Claim 3.3 we will get the same minor when running Algorithm 3.1 for either G or $\hat{G}$. Thus, in fact, we will execute Algorithm 3.1 on the original graph with no modifications.

4. Distortion analysis.

4.1. Interval and charges. In this section we describe in detail the probabilistic process of breaking the graph into clusters from the viewpoint of the Steiner vertices. The main objective will be to define a charging scheme, which we can later use to bound the distortion.

Consider two terminals t and t'. Let $P_{t,t'} = \{t = v_0, \ldots, v_\gamma = t'\}$ be the shortest path from t to t' in G. We can assume that there are no terminals in $P_{t,t'}$ other than t, t'. This is because if we will prove that for every pair of terminals t, t' such that $P_{t,t'} \cap K = \{t, t'\}$ it holds that $d_M(t, t') \leq O(\log k) \cdot d_G(t, t')$, this property will be implied for all terminal pairs.

For an interval $Q = \{v_a, \ldots, v_b\} \subseteq P_{t,t'}$, the *internal length* is $L(Q) = d_G(v_a, v_b)$, while the *external length* is $L^+(Q) = d_G(v_{a-1}, v_{b+1})$.[1] The distance from the interval Q to the terminals, denoted $D(Q) = D(v_a)$, is simply the distance from its leftmost point v_a to the closest terminal to v_a. Set $c_{\text{int}} = \frac{1}{6}$ ("int" for interval). We partition the vertices in $P_{t,t'}$ into consecutive intervals $\mathcal{Q}$ such that for every $Q \in \mathcal{Q}$,

$$L(Q) \leq c_{\text{int}}\delta \cdot D(Q) \leq L^+(Q) \ . \tag{4.1}$$

Such a partition could be constructed as follows. Sweep along the interval $P_{t,t'}$ in a greedy manner; after partitioning the prefix $v_0, \ldots, v_{h-1}$, to construct the next Q, simply pick the minimal index s such that $L^+(\{v_h, \ldots, v_{h+s}\}) \geq c_{\text{int}}\delta \cdot D(v_h)$. By the minimality of s, $L(\{v_h, \ldots, v_{h+s}\}) \leq L^+(\{v_h, \ldots, v_{h+s-1}\}) \leq c_{\text{int}}\delta \cdot D(v_h)$ (in the case $s = 0$, trivially $L(\{v_h\}) = 0 \leq c_{\text{int}}\delta \cdot D(v_h)$). Note that such s could always be found, as $L^+(\{v_h, \ldots, v_\gamma\}) = d_G(v_{h-1}, t') \geq d_G(v_h, t') \geq D(v_h) = D(Q)$.

In the beginning of Algorithm 3.1, all the vertices of $P_{t,t'}$ are *active*. Consider round j in the algorithm when terminal t_j constructs its cluster V_j. Specifically, it picks g_j and sets $R_j \leftarrow (1+\delta)^{g_j}$. Then, using the `Create-Cluster` procedure it grows a cluster in a "Dijkstra" fashion. If no active vertex joins V_j, we say that t_j doesn't *participate* in $P_{t,t'}$. Otherwise, let $a_j \in P_{t,t'}$ (resp., b_j) be the active vertex that joins to V_j with minimal (resp., maximal) index (w.r.t. $P_{t,t'}$). All the vertices $\{a_j, \ldots, b_j\} \subset P_{t,t'}$ between a_j and b_j (w.r.t. the order induced by $P_{t,t'}$) become inactive. We call this set $\{a_j, \ldots, b_j\}$ a *detour* $\mathcal{D}_j$ from a_j to b_j. See Figure 4.1 for an illustration.

Within each interval Q, each maximal subinterval of active vertices is called a *slice*. We denote by $\mathcal{S}(Q)$ the current number of slices in Q. In the beginning of the algorithm, for every interval Q, $\mathcal{S}(Q) = 1$, while at the end of the algorithm $\mathcal{S}(Q) = 0$.

For an active vertex v, let r_v be the minimal choice of R_j (determined by g_j) that will force v to join V_j. Let v^j be the active vertex with minimal r_v (breaking ties arbitrarily). Note that V_j is monotone with respect to R_j. That is, if v will join V_j for $R_j = r$, it will join V_j for $R_j = r' \geq r$ as well. We denote by $Q_j \in \mathcal{Q}$ the interval containing v^j. Similarly, S_j is the slice containing v^j. We *charge* Q_j for the detour $\mathcal{D}_j$. We denote by $X(Q)$ the number of detours the interval Q is currently charged for. For every detour $\mathcal{D}_{j'}$ which is contained in $\mathcal{D}_j$ (that is, $a_j < a_{j'} < b_{j'} < b_j$ w.r.t. the order induced by $P_{t,t'}$), we erase the detour and its charge. That is, for every

[1] For ease of notation we will denote $v_{-1} = t$ and $v_{\gamma+1} = t'$.

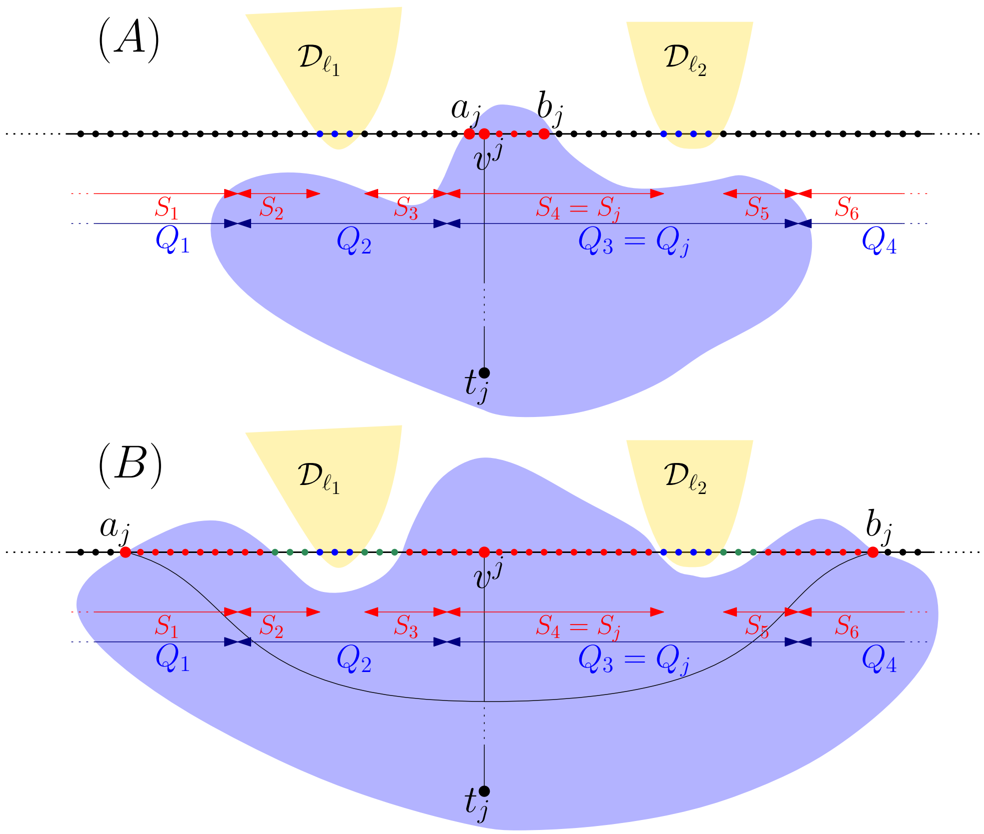

FIG. 4.1. *The figure illustrates round j in Algorithm* 3.1, *when t_j grows the cluster V_j. We present two scenarios for different choices of R_j. The black line is part of $P_{t,t'}$ the shortest path from t to t'. The blue intervals Q_i represent the intervals in $\mathcal{Q}$. The red subintervals S_i represent the slices (maximal continuous subsets of active vertices), where $S_2, S_3 \subset Q_2$ and $S_4, S_5 \subset Q_3$. The yellow areas represent detours $\mathcal{D}_{\ell_1}$ and $\mathcal{D}_{\ell_2}$, where Q_2 (resp., Q_3) is charged for $\mathcal{D}_{\ell_1}$ (resp., $\mathcal{D}_{\ell_2}$). Note that vertices in those areas are inactive. The terminal t_j increases gradually R_j, and the first vertex to be covered is v^j. In scenario* (A), *the growth of R_j terminates immediately after covering v^j and sets the borderline vertices a_j and b_j within the subinterval S_j. In scenario* (B), *the growth of R_j continues for another step, setting both a_j and b_j out of S_j. Vertices already inactive are shown in blue. Vertices that join the cluster V_j are shown in red. The green vertices are vertices which are still uncovered, but nevertheless become inactive. Vertices which remain active after the creation of V_j are colored in black. In scenario* (A) *all the vertices that become inactive, $\mathcal{D}_j$, are included in S_4. Q_3 is charged for $\mathcal{D}_j$. The number of slices in Q_3 is increased by* 1, *and no other changes occur ($X(Q_2) = 1$, $X(Q_3) = 2$). In scenario* (B) *$\mathcal{D}_\ell$ contains all the vertices in S_2, S_3, S_4, S_5 and part of the vertices in S_1, S_6. The number of slices in Q_2 and Q_3 becomes* 0, *while the number of slices in Q_1 and Q_4 remains unchanged. Q_3 is charged for $\mathcal{D}_\ell$, while its charge for $\mathcal{D}_{\ell_2}$ is erased. Additionally, the charge of Q_2 for $\mathcal{D}_{\ell_1}$ is erased. That is, Q_2 will remain uncharged till the end of the algorithm ($\tilde{X}(Q_2) = X(Q_2) = 0$, $X(Q_3) = 1$).*

$Q' \neq Q_j$, $X(Q')$ might only decrease, while $X(Q_j)$ might increase by at most 1 (and can also decrease as a result of deleted detours). We denote by $\tilde{X}(Q)$ the size of $X(Q)$ by the end of Algorithm 3.1. Figure 4.1 illustrates a single step.

Next, we analyze the change in the number of slices as a result of constructing the cluster V_j. If $R_j < r_{v^j}$, then no active vertex joins V_j and therefore $X(Q)$ and $\mathcal{S}(Q)$ stay unchanged, for all $Q \in \mathcal{Q}$. Otherwise, $R_j \geq r_{v^j}$, a new detour will appear and will be charged upon Q_j. All the slices S which are contained in $\mathcal{D}_j$ are deleted. Every slice S that intersects $\mathcal{D}_j$ but is not contained in it will be replaced by one or two new slices. If $\mathcal{D}_j \cap S \notin \{\mathcal{D}_j, S\}$, then S is replaced by a single new subslice S'. The only possibility for a slice to be replaced by two subslices is if $\mathcal{D}_j \subseteq S$, and $\mathcal{D}_j$ does not contain an "extremal" vertex in S (see Figure 4.1, scenario (A)). This can happen only at S_j. We conclude that for every $Q' \neq Q_j$, $\mathcal{S}(Q')$ might only decrease, while $\mathcal{S}(Q_j)$ might increase by at most 1.

CLAIM 4.1. *Assuming $R_j \geq r_{v^j}$, all of S_j joins V_j with probability at least $1-p$.*

Proof. As v^j joins V_j for $R_j \geq r_{v^j}$, by line 7 of Algorithm 3.2, necessarily $\frac{d_G(v^j,t_j)}{D(v^j)} \leq r_{v^j}$. We will argue that for every $u \in S_j$, the following inequality holds:

$$\frac{d_G(u,t_j)}{D(u)} \leq \frac{d_G(v^j,t_j)}{D(v^j)}(1+\delta) \leq r_{v^j}(1+\delta) \ . \tag{4.2}$$

Next, assume that $R_j \geq (1+\delta)r_{v^j}$. Before the execution of the `Create-Cluster` procedure for V_j, all the vertices in S_j belong to $V_\perp$ (as all of them are active). Because $R_j \geq r_{v^j}$, v^j will join V_j (by the definition of r_{v^j}). In particular, additional vertices from S_j (if they exist) will join N. Using inequality (4.2), for every $u \in S_j$, $d_G(u,t_j)/D_u \leq r_{v_j}(1+\delta) \leq R_j$. Therefore every vertex from S_j joining N will also join V_j. In such a way, since S_j is connected in $V_\perp$, all the vertices of S_j will join V_j, as required.

Next, we analyze the probability that indeed $R_j \geq (1+\delta)r_{v^j}$. Recall that $R_j = (1+\delta)^{g_j}$, where g_j is distributed according to geometric distribution with parameter $P_{t,t'}$. Conditioned on the event $R_j \geq r_{v^j}$, we have that

$$\begin{aligned} &\Pr\left[R_j \geq (1+\delta)r_{v^j} \mid R_j \geq r_{v^j}\right] \\ &= \Pr\left[g_j \geq \log_{1+\delta}\left((1+\delta)r_{v^j}\right) \mid g_j \geq \log_{1+\delta} r_{v^j}\right] \\ &= \Pr\left[g_j \geq 1+\log_{1+\delta} r_{v^j} \mid g_j \geq \log_{1+\delta} r_{v^j}\right] = 1-p \ . \end{aligned} \tag{4.3}$$

It remains to prove inequality (4.2). By the definition of $D(Q_j)$ and the triangle inequality

$$\begin{aligned} L(Q_j) &\overset{(4.1)}{\leq} c_{\text{int}}\delta \cdot D(Q_j) \leq c_{\text{int}}\delta \cdot \left(D(v^j) + L(Q_j)\right) \\ &\leq 2c_{\text{int}}\delta \cdot D(v^j) \leq 2c_{\text{int}}\delta \cdot d_G(v^j,t_j) \ . \end{aligned} \tag{4.4}$$

Therefore, for every $u \in S_j$,

$$d_G(u,t_j) \leq d_G(v^j,t_j) + L(Q_j) \overset{(4.4)}{\leq} d_G(v^j,t_j)\,(1+2c_{\text{int}}\delta) \,.$$

Similarly,

$$D(u) \geq D(v^j) - L(Q_j) \geq D(v^j)\,(1-2c_{\text{int}}\delta) \ . \tag{4.5}$$

We conclude that

$$\frac{d_G(u,t_j)}{D(u)} \leq \frac{d_G(v^j,t_j)\,(1+2c_{\text{int}}\delta)}{D(v^j)\,(1-2c_{\text{int}}\delta)} \leq \frac{d_G(v^j,t_j)}{D(v^j)}(1+3\cdot 2c_{\text{int}}\delta) = \frac{d_G(v^j,t_j)}{D(v^j)}(1+\delta) \ .$$

∎

4.2. Bounding the number of failures. Next, we define a *cost function* $f : \mathbb{R}_+^{|\mathcal{Q}|} \to \mathbb{R}_+$. Intuitively, the cost function is simply a summation over the intervals, where for each interval Q we add its length $L(Q)$ for each time it was charged. Formally, $f(\{x_Q\}_{Q\in\mathcal{Q}}) = \sum_{Q\in\mathcal{Q}} x_Q \cdot L^+(Q)$. Even though our goal will be to bound $f(\{\tilde{X}(Q)\}_{Q\in\mathcal{Q}})$, we define f as a general function from $\mathbb{R}^{|\mathcal{Q}|}$ in order to use it on other variables as well. Note that the cost function f is linear and monotonically increasing coordinatewise. In subsection 4.3 we show that the distance $d_M(t,t')$ between t and t' in the minor graph M can be bounded by $\log k \cdot f(\{\tilde{X}(Q)\}_{Q\in\mathcal{Q}})$, the scaled cost function applied on the charges. This section is devoted to proving the following lemma.

LEMMA 4.2. $\Pr[f(\{\tilde{X}(Q)\}_{Q\in\mathcal{Q}}) \geq 43 \cdot d_G(t,t')] \leq k^{-3}$.

Using Claim 4.1, one can show that for every $Q \in \mathcal{Q}$, $\mathbb{E}[\tilde{X}(Q)] = O(1)$, and moreover, w.h.p. $\tilde{X}(Q) = O(\log k)$ for all Q. However, we use a concentration bound on all $\{\tilde{X}(Q)\}_{Q\in\mathcal{Q}}$ simultaneously in order to provide a stronger upper bound.

4.2.1. Bounding by independent variables. In our journey to bound $f(\{\tilde{X}(Q)\}_{Q\in\mathcal{Q}})$, the first step will be to replace $\{\tilde{X}(Q)\}_{Q\in\mathcal{Q}}$ with independent random variables. Consider the following process: a *box* B which contains coins of two types, active and inactive. In the beginning, there is a single active coin. In each round, we toss an active coin, which gets 0 (failure) with probability p, and 1 (success) with probability $1-p$. If we get a 0, two additional active coins are added to the box. In any case, the tossed coin becomes inactive. All the coin tosses throughout the process are independent. The process terminates when no active coins remain. Let $\{B_Q\}_{Q\in\mathcal{Q}}$ be a set of $|\mathcal{Q}|$ independent boxes (here the box B_Q resembles the interval Q). For the box B_Q, denote by $Z(Q)$ the number of active coins, by $Y(Q)$ the number of inactive coins, and by $\tilde{Y}(Q)$ the number of inactive coin at the end of the process.

CLAIM 4.3. *For every* $\alpha \in \mathbb{R}_+$,

$$\Pr\left[f\left(\{\tilde{X}(Q)\}_{Q\in\mathcal{Q}}\right) \geq \alpha\right] \leq \Pr\left[f\left(\{\tilde{Y}(Q)\}_{Q\in\mathcal{Q}}\right) \geq \alpha\right] \ .$$

Proof. The proof is done by coupling the two processes of Algorithm 3.1 and the coin tosses. We execute Algorithm 3.1, which implicitly induces slices and detour charges. Simultaneously, we will use Algorithm 3.1 to toss coins. Inductively, we will maintain the invariant that $\{Y(Q)\}_{Q\in\mathcal{Q}}$ and $\{Z(Q)\}_{Q\in\mathcal{Q}}$ are no less than $\{X(Q)\}_{Q\in\mathcal{Q}}$ and $\{S(Q)\}_{Q\in\mathcal{Q}}$ (respectively) coordinatewise.

In the beginning $\{X(Q)\}_{Q\in\mathcal{Q}} = \{Y(Q)\}_{Q\in\mathcal{Q}} = \{0\}_{Q\in\mathcal{Q}}$ and $\{S(Q)\}_{Q\in\mathcal{Q}} = \{Z(Q)\}_{Q\in\mathcal{Q}} = \{1\}_{Q\in\mathcal{Q}}$. Consider round j, where the cluster V_j is created for the terminal t_j. If $R_j < r_{v^j}$, then nothing happens, and the invariant holds. Else, $R_j \geq r_{v^j}$, we will make a coin toss from the B_{Q_j} box. Let p' be the probability that not all of S_j joins V_j. By Claim 4.1, $p' \leq p$. If indeed not all of S_j joins V_j, the toss result is set to 0. Otherwise, with probability $\frac{p-p'}{1-p'}$ the toss is set to 0. Note that the probability of 0 is exactly $p' \cdot 1 + (1-p') \cdot \frac{p-p'}{1-p'} = p$.

Next we argue that the invariant is maintained in either case. If not all of S_j joins Q_j, then $S(Q_j)$ might increase by at most one, while the number of active coins Z_{Q_j} increases by exactly one. Otherwise, all of S_j joins Q_j. In this case $S(Q_j)$ necessarily decreases by at least one, while Z_{Q_j} might either decrease or increase by one. For the charge parameter, $X(Q_j)$ might increase by at most one, while the number of inactive coins $Y(Q_j)$ increases by exactly one. For every $Q' \neq Q_j$, $\mathcal{S}(Q')$ and $X(Q')$ might only decrease, while $Z_{Q'}$ and $Y(Q')$ stay unchanged. We conclude that the invariant holds after the construction of the cluster V_j.

Intuitively speaking, creating a cluster for a terminal t_j is a global processes that can involve many slices in different terminals, the crux being that only the interval Q_j is charged, and only the slice S_j might get splitted. For all other intervals, charges can only get erased and slices eliminated. The process of coin tosses in the boxes imitates charge and slice counting, while ignoring the potential savings.

At the end of the algorithm (when no slices are left), we might still have some active coins. In this case we will simply toss coins until no active coins remain (note that this indeed happens with probability 1). Note that by doing so $\{Y(Q)\}_{Q\in\mathcal{Q}}$ can only grow coordinatewise. As the marginal distribution on $\{\tilde{Y}(Q)\}_{Q\in\mathcal{Q}}$ is exactly identical to the original one, the claim follows. ∎

4.2.2. Replacing coins with exponential random variables. Our next step is to replace each $Y(Q)$ with exponential random variable. This replacement will make the use of concentration bounds more convenient. Consider some box B_Q. An equivalent way to describe the probabilistic process in B_Q is the following. Take a single coin with failure probability p, and toss this coin until the number of successes exceeds the number of failures. The total number of tosses is exactly $\tilde{Y}(Q)$. Note that $\tilde{Y}(Q)$ is necessarily odd. Next we bound the probability that $\tilde{Y}(Q)\ge 2m+1$ for $m\ge 1$. This is obviously upper bounded by the probability that in a series of $2m$ tosses we had at least m failures (as otherwise the process would have stopped earlier). Let χ_i be an indicator for a failure in the ith toss, and $\chi=\sum_{i=1}^{2m}\chi_i$. Note that $\mathbb{E}[\chi]=2m\cdot p$. A bound on χ follows by the Chernoff inequality.

Fact 1 (Chernoff inequality). Let $X_1,\dots,X_n$ be independent and identically distributed (i.i.d.) indicator variables each with probability p. Set $X=\sum_i X_i$ and $\mu=\mathbb{E}[X]=np$. Then for every $\delta\le 2e-1$, $\Pr[X\ge(1+\delta)\mu]\le\exp(-\mu\delta^2/4)$.

$$\begin{aligned}\Pr\left[\tilde{Y}(Q)\ge 2m+1\right]&\le\Pr[\chi\ge m]=\Pr\left[\chi\ge\left(1+\left(\frac{1}{2p}-1\right)\right)\mathbb{E}[\chi]\right]\\&\le\exp\left(-2m\cdot p\cdot\left(\frac{1}{2p}-1\right)^2/4\right)=\exp\left(-\frac{9}{40}m\right)\\&\le\exp\left(-\frac{1}{5}m\right)\ .\end{aligned}$$

We conclude that the distribution of $\tilde{Y}(Q)$ is dominated by $1+\mathsf{Exp}(10)$ (as for $W\sim\mathsf{Exp}(10)$, $\Pr[1+W\ge 2m+1]=\exp\left(-\frac{m}{5}\right)$). Let $(\{W(Q)\}_{Q\in\mathcal{Q}})$ be i.i.d. random variables distributed according to $\mathsf{Exp}(10)$; since all the boxes are independent and f is linear and monotone coordinatewise, we conclude as follows.

CLAIM 4.4. *For every $\alpha\in\mathbb{R}_+$,*

$$\Pr\left[f\left(\left\{\tilde{Y}(Q)\right\}_{Q\in\mathcal{Q}}\right)\ge\alpha\right]\le\Pr\left[f\left(\{1\}_{Q\in\mathcal{Q}}\right)+f\left(\{W(Q)\}_{Q\in\mathcal{Q}}\right)\ge\alpha\right].$$

Proof. Set $\varphi=|\mathcal{Q}|$. Let $Q^1,Q^2,\dots,Q^\varphi$ be some arbitrarily fixed ordering of the intervals. For $s\in[\varphi]$, set $f_{\setminus\{s\}}(x_1,\dots,x_{s-1},x_{s+1},\dots,x_\varphi)=\sum_{i\in[\varphi]\setminus\{s\}}x_i\cdot L^+(Q^i)$. When integrating over the appropriate measure space, it holds that

$$\begin{aligned}
&\Pr\left[f\left(\tilde{Y}(Q^1),\ldots,\tilde{Y}(Q^\varphi)\right)\geq\alpha\right]\\
&=\int_\beta \Pr\left[f_{\setminus\{1\}}\left(\tilde{Y}(Q^2),\ldots,\tilde{Y}(Q^\varphi)\right)=\beta\right]\\
&\qquad\qquad\cdot\Pr\left[\tilde{Y}(Q^1)\cdot L^+(Q^1)\geq\alpha-\beta\right]d\beta\\
&\leq\int_\beta \Pr\left[f_{\setminus\{1\}}\left(\tilde{Y}(Q^2),\ldots,\tilde{Y}(Q^\varphi)\right)=\beta\right]\\
&\qquad\qquad\cdot\Pr\left[\left(1+W(Q^1)\right)\cdot L^+(Q^1)\geq\alpha-\beta\right]d\beta\\
&=\Pr\left[f\left(1+W(Q^1),\tilde{Y}(Q^2),\ldots,\tilde{Y}(Q^\varphi)\right)\geq\alpha\right]\\
&\leq\Pr\left[f\left(1+W(Q^1),1+W(Q^2),\tilde{Y}(Q^3),\ldots,\tilde{Y}(Q^\varphi)\right)\geq\alpha\right]\\
&\leq\cdots\leq\Pr\left[f\left(1+W(Q^1),\ldots,1+W(Q^\varphi)\right)\geq\alpha\right]\\
&=\Pr\left[f\left(1,\ldots,1\right)+f\left(W(Q^1),\ldots,W(Q^\varphi)\right)\geq\alpha\right]\ . \qquad\square
\end{aligned}$$

4.2.3. Concentration. Set $\Delta = d_G(t,t')$. It holds that

$$\Delta \leq \sum_{Q\in\mathcal{Q}} L^+(Q) \leq 2\Delta\ ,$$

as every edge in $P_{t,t'}$ is counted at least once, and at most twice in this sum. In particular $f(\{1\}_{Q\in\mathcal{Q}})\leq 2\Delta$. Recall that by our modification step, every edge in $P_{t,t'}$ is of weight at most $c_w\cdot\Delta$. In particular, for every $Q\in\mathcal{Q}$, $L^+(\mathcal{Q})\leq L(\mathcal{Q})+2c_w\cdot\Delta$. For every vertex v on $P_{t,t'}$, it holds that $D(v)\leq\min\{d_G(v,t),d_G(v,t')\}\leq\frac{\Delta}{2}$. Therefore for every $Q\in\mathcal{Q}$,

$$L^+(\mathcal{Q})\leq L(\mathcal{Q})+2c_w\cdot\Delta\overset{(4.1)}{\leq}c_{\text{int}}\delta\cdot D(Q)+2c_w\cdot\Delta\leq\left(\frac{c_{\text{int}}\delta}{2}+2c_w\right)\cdot\Delta=c_{\text{int}}\delta\cdot\Delta\ .$$

Let $\tilde{W}(Q)\sim L^+(Q)\cdot\mathsf{Exp}\,(10)$. In particular, $\tilde{W}(Q)\sim\mathsf{Exp}\,(10\cdot L^+(Q))$. Set $\tilde{W}=\sum_{Q\in\mathcal{Q}}\tilde{W}(Q)$. Then $f(\{W(Q)\}_{Q\in\mathcal{Q}})$ is distributed exactly as $\tilde{W}$. The maximal mean among the $\tilde{W}(Q)$'s is $\lambda_M=\max_{Q\in\mathcal{Q}}10\cdot L^+(Q)\leq 10\cdot c_{\text{int}}\delta\cdot\Delta$. The mean of $\tilde{W}$ is $\mu=\sum_{Q\in\mathcal{Q}}10\cdot L^+(Q)\leq 20\Delta$. Set $c_{\text{con}}=\frac{1}{2}$ (con for concentration). Using Claim 4.3, Claim 4.4, and Lemma 2.1, we conclude

$$\begin{aligned}
&\Pr\left[f\left(\left\{\tilde{X}(Q)\right\}_{Q\in\mathcal{Q}}\right)\geq(c_{\text{con}}+42)\Delta\right]\\
&\leq\Pr\left[f\left(\left\{\tilde{Y}(Q)\right\}_{Q\in\mathcal{Q}}\right)\geq(c_{\text{con}}+42)\Delta\right]\\
&\leq\Pr\left[f\left(\{W(Q)\}_{Q\in\mathcal{Q}}\right)\geq(c_{\text{con}}+42)\Delta-f\left(\{1\}_{Q\in\mathcal{Q}}\right)\right]\\
&\leq\Pr\left[\tilde{W}\geq(c_{\text{con}}+40)\Delta\right]\\
&\leq\exp\left(-\frac{1}{2\lambda_M}\left((c_{\text{con}}+40)\,\Delta-2\mu\right)\right)\\
&\leq\exp\left(-\frac{1}{2}\cdot\frac{1}{10c_{\text{int}}\delta\Delta}\cdot c_{\text{con}}\Delta\right)=\exp\left(-\frac{c_{\text{con}}}{20\cdot c_{\text{int}}\delta}\right)=k^{-3}\ .
\end{aligned}$$

Note that $c_{\text{con}}\leq 1$, thus Lemma 4.2 follows.

4.3. Bounding the distortion. Denote by $\mathcal{E}^{\text{fBig}}$ the event that for some pair of terminals t, t', $f(\{\tilde{X}(Q)\}_{Q\in\mathcal{Q}}) \geq 43 \cdot d_G(t,t')$.[2] By Lemma 4.2 and the union bound, $\Pr\left[\mathcal{E}^{\text{fBig}}\right] \leq \binom{k}{2} \cdot k^{-3} < \frac{1}{2k}$.

Let $\mathcal{E}^{\text{B}}$ be the event that for some j, $R_j > c_d$, where $c_d = e^2$. Note that if $\mathcal{E}^{\text{B}}$ does not hold, then every vertex v joins to a cluster V_j such that $d_G(v,t_j) \leq c_d \cdot D(v)$.

Claim 4.5. $\Pr[\mathcal{E}^{B}] \leq \frac{1}{2k}$.

Proof. Let $\mathcal{E}_j^{\text{B}}$ be the event that $R_j > c_d$. It holds that

$$\Pr[\mathcal{E}_j^{\text{B}}] = \Pr[g_j \geq \log_{1+\delta} c_d] \leq (1-p)^{\log_{1+\delta} c_d - 1} \leq (1-p)^{\frac{2}{\delta}-1} \leq \frac{1}{k^3} \ ,$$

where the second inequality holds as $\log_{1+\delta} c_d = \frac{\ln c_d}{\ln 1+\delta} \geq \frac{2}{\delta}$. By the union bound, $\Pr[\mathcal{E}^{\text{B}}] \leq \frac{1}{k^2} \leq \frac{1}{2k}$ as required. ∎

Lemma 4.6. *Assuming $\overline{\mathcal{E}^{B}}$ and $\overline{\mathcal{E}^{fBig}}$, for every pair of terminals t, t', $d_M(t,t') \leq O(\log k) \cdot d_G(t,t')$.*

Proof. Fix some t, t'. By the end of Algorithm 3.1, all the vertices in $P_{t,t'} = \{t = v_0, \ldots, v_\gamma = t'\}$ are divided into consecutive detours[3] $\mathcal{D}_{\ell_1}, \ldots, \mathcal{D}_{\ell_{k'}}$. The detour $\mathcal{D}_{\ell_j}$ was constructed at round ℓ_j by the terminal t_{ℓ_j}. The detour $\mathcal{D}_{\ell_j}$ was charged upon the interval Q_{ℓ_j}, which contains the vertex v^{ℓ_j}. The leftmost vertex in $\mathcal{D}_{\ell_j}$ is called a_{ℓ_j}, while the rightmost vertex is called b_{ℓ_j}. In particular, for every $1 \leq j \leq k'-1$, there is an edge in G between b_{ℓ_j} and $a_{\ell_{j+1}}$, and therefore there is an edge between t_{ℓ_j} to $t_{\ell_{j+1}}$ in the terminal-centered minor M. As $t = v_0$ joins the cluster of itself, necessarily $t_{\ell_1} = t$. Similarly $t_{\ell_{k'}} = t'$. See Figure 4.2 for an illustration. Using the triangle inequality, we conclude

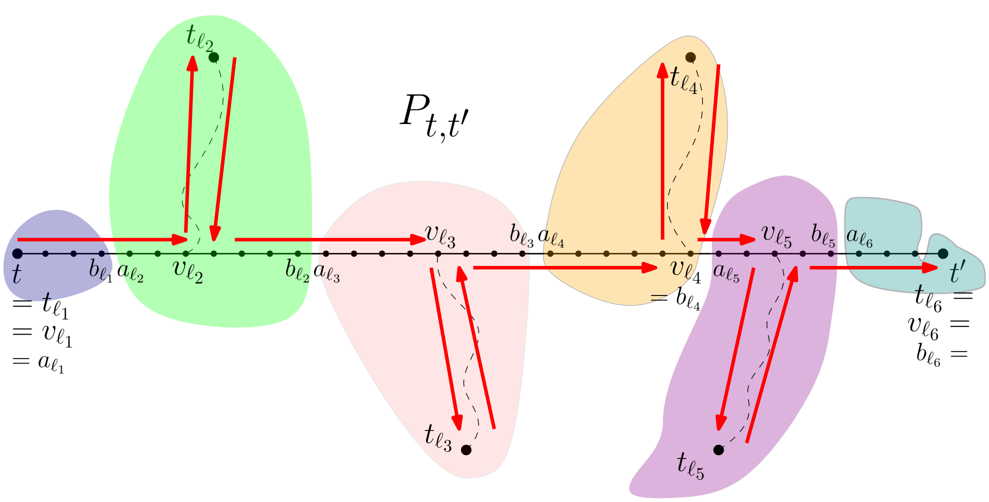

Fig. 4.2. *The vertices $P_{t,t'} = v_0 \ldots v_\gamma$ are divided into consecutive detours $\mathcal{D}_{\ell_1}, \ldots, \mathcal{D}_{\ell_6}$. $t_{\ell_1}, t_{\ell_2}, t_{\ell_3}, t_{\ell_4}, t_{\ell_5}, t_{\ell_6}$ is a path in the terminal-centered minor M of G (induced by $V_1, \ldots, V_k$). The weight of the edge $\{t_{\ell_j}, t_{\ell_{j+1}}\}$ in M is $d_G(t_{\ell_j}, t_{\ell_{j+1}})$, which is bounded by $d_G(t_{\ell_j}, v_{\ell_j}) + d_G(v_{\ell_j}, v_{\ell_{j+1}}) + d_G(v_{\ell_{j+1}}, t_{\ell_{j+1}})$.*

[2] We abuse notation here and use the same $\{\tilde{X}(Q)\}_{Q\in\mathcal{Q}}$ for all terminals.

[3] Note that we consider only detours that inflict a charge by the end of the algorithm. Therefore the detours are disjoint and every vertex in $P_{t,t'}$ belongs to some detour.

$$d_M(t,t') \le \sum_{j=1}^{k'-1} d_G(t_{\ell_j}, t_{\ell_{j+1}}) \le \sum_{j=1}^{k'-1} \left[d_G(t_{\ell_j}, v^{\ell_j}) + d_G(v^{\ell_j}, v^{\ell_{j+1}}) + d_G(v^{\ell_{j+1}}, t_{\ell_{j+1}}) \right]$$
$$\le \sum_{j=1}^{k'-1} d_G(v^{\ell_j}, v^{\ell_{j+1}}) + 2\sum_{j=1}^{k'} d_G(t_{\ell_j}, v^{\ell_j})$$
$$\le d_G(t,t') + 2\sum_{j=1}^{k'} c_d \cdot D(v^{\ell_j}),$$

where the last inequality follows by our assumption $\overline{\mathcal{E}^{\mathrm{B}}}$. By the definition of $D(Q_{\ell_j})$, inequality (4.1) and the triangle inequality, $D(v^{\ell_j}) \le D(Q_{\ell_j}) + L(Q_{\ell_j}) \le (\frac{1}{c_{\mathrm{int}}\delta} + 1)$ $L^+(Q_{\ell_j}) \le \frac{2}{c_{\mathrm{int}}\delta} \cdot L^+(Q_{\ell_j})$. Using the assumption $\overline{\mathcal{E}^{\mathrm{fBig}}}$, we conclude

$$\begin{aligned} (4.6) \qquad d_M(t,t') &\le d_G(t,t') + 2c_d \sum_{i=1}^{k'} \frac{2}{c_{\mathrm{int}}\delta} \cdot L^+(Q_{\ell_i}) \\ &= d_G(t,t') + \frac{4c_d}{c_{\mathrm{int}}\delta} \sum_{Q\in\mathcal{Q}} \tilde{X}(Q) \cdot L^+(Q) \\ &= d_G(t,t') + \frac{4c_d}{c_{\mathrm{int}}\delta} \cdot f\left(\{\tilde{X}(Q)\}_{Q\in\mathcal{Q}}\right) = O(\ln k) \cdot d_G(t,t') \ . \end{aligned}$$

As $\Pr\left[\overline{\mathcal{E}^{\mathrm{B}}} \wedge \overline{\mathcal{E}^{\mathrm{fBig}}}\right] \ge 1 - (\Pr[\mathcal{E}^{\mathrm{B}}] + \Pr[\mathcal{E}^{\mathrm{fBig}}]) \ge 1 - \frac{1}{2k} - \frac{1}{2k} = 1 - \frac{1}{k}$, Theorem 3.1 follows. ∎

5. `Fast-Relaxed-Voronoi` algorithm. In this section, we describe a slightly modified version of the `Relaxed-Voronoi` algorithm. Then we will show how to implement the modified algorithm in $O(m \log n)$ time.

Given two terminals t_i, t_j, and two clusters $V_i, V_j \subseteq V$ s.t. t_i (resp., t_j) is the unique terminal in V_i (resp., V_j), $d_{G,V_i+V_j}(t_i,t_j)$ denotes the length of the shortest path between t_i and t_j in $G[V_i \cup V_j]$ that uses exactly one crossing edge between V_i and V_j. See Figure 5.1 for an illustration.

In order to allow fast implementation, and avoid costly shortest path computations, we will introduce several modifications:

- In Algorithm 3.1, line 9, we will modify the edge weights in the induced terminal-centered minor. The weight of the edge $\{t_i,t_j\}$ (if exists) will be $d_{G,V_i+V_j}(t_i,t_j)$ instead of $d_G(t_i,t_j)$.

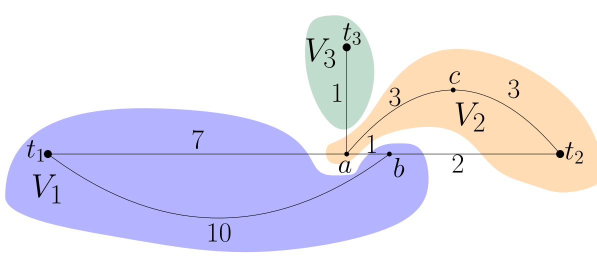

FIG. 5.1. *t_1, t_2, t_3 are terminals. The different color areas describes the terminal partition. The shortest path in G from t_1 to t_2 is t_1, a, b, t_2 and has length $d_G(t_1,t_2) = 10$. Note that all the vertices in this path are in $V_1 \cup V_2$. Nevertheless, the shortest path from t_1 to t_2 that uses only one crossing edge from t_1 to t_2 is $\{t_1, b, t_2\}$ and has length $d_{G,V_1+V_2}(t_1,t_2) = 12$.*

Algorithm 5.1. $M =$ `Fast-Relaxed-Voronoi`$(G = (V, E, w), K = \{t_1, \ldots, t_k\})$.

Set $\delta = \frac{1}{20 \ln k}$ and $p = \frac{1}{5}$.
Set $V_\perp \leftarrow V \setminus K$. *// $V_\perp$ is the currently unclustered vertices.*
for j from 1 to k **do**
  Choose independently at random g_j distributed according to $\mathsf{Geo}(p)$.
  Set $R_j \leftarrow (1+\delta)^{g_j}$.
  Set $V_j \leftarrow$ `Fast-Create-Cluster`$(G, V_\perp, t_j, R_j)$.
  Remove all the vertices in V_j from $V_\perp$.
end for
Let M be the minor of G created by contracting all the internal edges in $V_1, \ldots, V_k$. The weight of the edge $\{t_i, t_j\}$ (if it exists) is defined to be $d_{G, V_i + V_j}(t_i, t_j)$.
return M.

Algorithm 5.2. $V_j =$ `Fast-Create-Cluster`$(G = (V, E, w), V_\perp, t_j, R_j)$.

Set $V_j \leftarrow \{t_j\}$.
Set $U \leftarrow \emptyset$. *// U is the set of vertices already denied from V_j.*
Set N to be all the neighbors of t_j in $V_\perp$.
while $N \neq \emptyset$ **do**
  Let $v \in N$ be the vertex with minimal $d_{G[V_j \cup \{v\}]}(v, t_j)$.
  Remove v from N.
  if $d_{G[V_j \cup \{v\}]}(v, t_j) \leq R_j \cdot D(v)$ **then**
    Add v to V_j.
    Add all the neighbors of v in $V_\perp \setminus U$ to N.
  else
    Add v to U.
  end if
end while
return V_j.

- In Algorithm 3.2, line 5, instead of extracting an arbitrary vertex v from N, we will extract the closest vertex v to t_j in N w.r.t. the shortest path metric induced by $V_j \cup \{v\}$ (i.e., $v \in N$ with minimal $d_{G[V_j \cup \{v\}]}(v, t_j)$, and note that it is a different graph for each vertex).
  Similarly, in line 7, instead of checking whether $d_G(v, t_j) \leq R_j \cdot D(v)$, we will check whether $d_{G[V_j \cup \{v\}]}(v, t_j) \leq R_j \cdot D(v)$.

The pseudocode of the modified algorithm appears in Algorithms 5.1 and 5.2.

THEOREM 5.1. *With probability $1 - \frac{1}{k}$, for the minor graph M returned by Algorithm* 5.1, *it holds that for every two terminals t, t', $d_M(t, t') \leq O(\log k) \cdot d_G(t, t')$. Moreover, executing Algorithm* 5.1 *takes $O(m + \min\{m, nk\} \cdot \log n)$ time.*

We prove Theorem 5.1 in several steps. First, in subsection 5.1 we show that Algorithm 5.1 indeed returns a terminal partition and that similarly to Algorithm 3.1, the edge subdivision does not change the outcome of the algorithm. Then in subsection 5.2 we'll go through the analysis provided in section 4 and verify that it still goes through for Algorithm 5.1 as well. Finally, in subsection 5.3 we describe an efficient implementation of Algorithm 5.1.

5.1. Basic properties. Consider the `Fast-Create-Cluster` procedure (Algorithm 5.2). This is a Dijkstra-like algorithm. For every vertex v, set $\ell_v = d_{G[V_j\cup\{v\}]}(v, t_j)$. Note that for a vertex v, the value ℓ_v is decreasing throughout the algorithm as the set V_j grows. Note also that ℓ_v is defined for all the vertices (but simply has value ∞ for vertices out of $V_j \cup N$). Denote by $\hat{\ell}_v$ the value ℓ_v at the time v is extracted from N at line 6 of Algorithm 5.2 (if such an occasion indeed occurs).

Claim 5.2. *Consider the values $\hat{\ell}_v$ of the vertices, extracted from N at line* 6 *of Algorithm* 5.2*. Then these values are nondecreasing. That is, if v was extracted before v', then $\hat{\ell}_v \le \hat{\ell}_{v'}$.*

Moreover, after v is extracted, the value ℓ_v remains unchanged till the end of the algorithm.

Proof. The proof of the first property is by induction on the execution of the algorithm. Let v, v' be a pair of vertices such that v' was extracted from N right after v. It will be enough to show that $\hat{\ell}_v \le \hat{\ell}_{v'}$. Consider the time when v was extracted from N. Let $\tilde{V}_j$ denote the set V_j at that time. By minimality, for every $u \in N$, $\hat{\ell}_v = d_{G[\tilde{V}_j\cup\{v\}]}(v, t_j) \le d_{G[\tilde{V}_j\cup\{u\}]}(u, t_j)$. If the value $\ell_{v'}$ did not change, we already have $\hat{\ell}_{v'} = d_{G[\tilde{V}_j\cup\{v'\}]}(v', t_j) \ge \hat{\ell}_v$ (as necessarily $v' \in N$ because it is extracted next). Otherwise, if the value $\ell_{v'}$ decreased, then necessarily v joined V_j and the shortest path from from t_j to v' (in $\tilde{V}_j \cup \{v, v'\}$) goes through v (as otherwise $\ell_{v'}$ would not have changed). In particular, $\hat{\ell}_{v'} = d_{G[\tilde{V}_j\cup\{v,v'\}]}(t_j, v') = d_{G[\tilde{V}_j\cup\{v,v'\}]}(t_j, v) + d_{G[\tilde{V}_j\cup\{v,v'\}]}(v, v') > \hat{\ell}_v$.

For the second property (that after extraction, ℓ_v remains unchanged), seeking contradiction, assume that ℓ_v is updated after some u is extracted from N and joined V_j. This implies that the new shortest path from t_j to v goes through u and thus is of length greater than $\hat{\ell}_u$, a contradiction. ∎

Now we are ready to show that Algorithm 5.1 indeed returns a terminal partition (that is, reprove Lemma 3.2).

Lemma 5.3. *The sets $V_1, \ldots, V_k$ constructed by Algorithm* 5.1 *form a terminal partition.*

Proof. It is clear that the clusters $V_1, \ldots, V_j$ are disjoint and that each cluster is connected. It will be enough to argue that every vertex $v \in V$ is clustered. Following along the lines of the proof of Lemma 3.2, let t_j be the closest terminal to v, and let $P = \{t_j = u_0, u_1, \ldots, u_s = v\}$ be the shortest path from t_j to v. Let $u_{i'}$ be the first vertex from $P_{t,t'}$ to be clustered during the algorithm ($u_0 = t_j \in V_j$, so at least one vertex in $P_{t,t'}$ is clustered). Let $V_{j'}$ be the cluster $u_{i'}$ joins to. We argue by induction on $i \ge i'$ that u_i also joins $V_{j'}$. This will imply that $u_s = v$ joins $V_{j'}$ and thus is clustered.

Suppose u_i joins $V_{j'}$. Denote by $V^i_{j'}$ the set $V_{j'}$ right after u_i joins it. As u_i joins $V_{j'}$, $d_{G[V^i_{j'}]}(u_i, t_{j'}) \le R_{j'} \cdot D(u_i)$. In particular, at that stage

$$\begin{aligned}\ell_{u_{i+1}} = d_{G\left[V^i_{j'}\cup\{u_{i+1}\}\right]}(u_{i+1}, t_{j'}) &\le d_{G\left[V^i_{j'}\right]}(u_i, t_{j'}) + w\left(\{u_i, u_{i+1}\}\right)\\ &\le R_{j'} \cdot D(u_i) + d_G(u_i, u_{i+1}) \le R_{j'} \cdot D(u_{i+1}).\end{aligned}$$

As at least one neighbor (u_i) of u_{i+1} joins $V_{j'}$, u_{i+1} joins N at some stage of the algorithm. In particular, by Claim 5.2, when u_{i+1} will be extracted from N, $\hat{\ell}_{u_{i+1}} \leq R_{j'} \cdot D(u_{i+1})$, and thus u_{i+1} will join $V_{j'}$ as required. ∎

We will use the modified graph $\hat{G}$ (with the subdivided edges) for the distortion analysis. In order to prove validity, we will argue that Claim 3.3 still holds.

CLAIM 5.4. *In Claim* 3.3*, if we replace Algorithm* 3.1 *with Algorithm* 5.1*, the claim still holds.*

Proof. We follow the lines of the proof of Claim 3.3. Let $V_1, \dots, V_k$ (resp., $\tilde{V}_1, \dots, \tilde{V}_k$) be the terminal partition induced by Algorithm 5.1 on G (resp., $\tilde{G}$). We argue that for all j, $V_j = \tilde{V}_j \setminus \{v_e\}$. As previously, this will imply that the terminal-centered minors have the same edges set. As v_e only subdivides the edge e, it will also hold for all i, j that $d_{G,V_i+V_j}(t_i, t_j) = d_{G,\tilde{V}_i+\tilde{V}_j}(t_i, t_j)$, and thus the edge weights in both minors will also be identical. In particular, the claim will follow.

Suppose w.l.o.g. that v joins V_j while u is still unclustered. Denote by V'_j (resp., $\tilde{V}'_j$) the set V_j (resp., $\tilde{V}_j$) right after the clustering of v at the execution of Algorithm 5.1 on G (resp., $\tilde{G}$). As previously, for all $j'' < j$, $V_{j''} = \tilde{V}_{j''}$, while $V'_j = \tilde{V}'_j$.

Recall that $\hat{\ell}_v = d_{G[V'_j](t_j,v)}$ (resp., $\tilde{\hat{\ell}}_v$) denotes the distance between t_j to v at the time of the extraction of v from N (resp. $\tilde{N}$). Note that $\hat{\ell}_v = \tilde{\hat{\ell}}_v$. As v joins V_j, necessarily $\hat{\ell}_v \leq R_j \cdot D(v)$. In the rest of the proof we consider the following cases:

- $\hat{\ell}_u > R_j \cdot D(v)$: In this case u will not join V_j. As v_e has edges only to v and u, v_e has no impact on any other vertex. In particular, $\hat{\ell}_u \leq \tilde{\hat{\ell}}_u$. Therefore $\tilde{V}_j$ will be constructed in the same manner as V_j (up to maybe containing v_e). Note that all the other clusters will not be affected, as if v_e remained unclustered, it becomes a leaf. We conclude that for every j', $V_{j'} = \tilde{V}_{j'} \setminus \{v_u\}$.
- $\hat{\ell}_u \leq R_j \cdot D(v)$: Recall that ω is the weight of e. There are two subcases:
  - $\hat{\ell}_u = \hat{\ell}_v + \omega$. After v joins $\tilde{V}_j$, the label of v_e is updated to $\hat{\ell}_{v_e} \leftarrow \tilde{\hat{\ell}}_v + \frac{\omega}{2}$. It holds that

$$\begin{aligned}\tilde{\hat{\ell}}_{v_e} \leq \tilde{\ell}_{v_e} = \tilde{\hat{\ell}}_v + \frac{\omega}{2} = \hat{\ell}_v + \frac{\omega}{2} = \frac{1}{2}\left(\hat{\ell}_v + \hat{\ell}_u\right) \\ \leq \frac{1}{2} \cdot R_j \left(D(v) + D(u)\right) \leq R_j \cdot D(e_v) \ .\end{aligned}$$

    In particular, v_e will join $\tilde{V}_j$, and $\tilde{\ell}_u$ will be updated to $\tilde{\hat{\ell}}_{v_e} + \frac{\omega}{2} = \tilde{\hat{\ell}}_v + \omega$. From this point on, the two algorithms will behave in the same way. In particular, for every $j'' \neq j$, $V_{j''} = \tilde{V}_{j''}$ while $V_j \cup \{v_e\} = \tilde{V}_j$.
  - $\hat{\ell}_u < \hat{\ell}_v + \omega$. It holds that u joins V_j. However, the shortest path in V_j from t_j to u did not goes through v. Therefore, as v_e did not affect any vertex (other than v, u), the execution will proceed in the same way in both algorithms, and u will join $\tilde{V}_j$. As each cluster is connected and all the vertices are clustered, necessarily v_e will join $\tilde{V}_j$ as well. We conclude that for every $j'' \neq j$, $V_{j''} = \tilde{V}_{j''}$ while $V_j \cup \{v_e\} = \tilde{V}_j$. ∎

5.2. Distortion analysis. We will follow the distortion analysis of Algorithm 3.1 given in section 4. Consider two terminals t, t'. We will use the exact same notation (the reader is referred to Appendix C in order to recall notation and definitions). We start by reproving Claim 4.1.

CLAIM 5.5. *During the execution of Algorithm* 5.1, *assuming* $R_j \geq r_{v^j}$, *all of* S_j *joins* V_j *with probability at least* $1-p$.

Proof. Denote $S_j = \{u_{j-q'}, \dots, u_j, \dots, u_{j+q}\} \subseteq Q_j \subseteq P_{t,t'}$ where $v^j = u_j$. Denote by V_j' the cluster V_j right after u_j joins. As u_j joined, necessarily $\frac{d_{G[V_j' \cup \{u_j\}]}(u_j,t_j)}{D(u_j)} \leq r_{v^j} \leq R_j$. We will denote by $\bar{V}_j$ the cluster V_j at the end of the algorithm. Following inequality (4.3), with probability $1-p$, $R_j \geq (1+\delta) r_{v^j}$. We will show that if this event indeed occurs, then $S_j \subseteq \bar{V}_j$.

We argue by induction on i that $u_{j+i} \in \bar{V}_j$. The proof that $u_{j-i} \in \bar{V}_j$ is symmetric. Assume that $\{u_i, u_{i+1}, \dots, u_{j+i-1}\} \subseteq \bar{V}_j$. Following inequalities (4.4) and (4.5), $L(Q_j) \leq 2c_{\text{int}}\delta \cdot D(v^j)$ and $D(u_{j+i}) \geq D(v^j)\,(1-2c_{\text{int}}\delta)$. As $u_{i+j-1} \in \bar{V}_j$, u_{j+i} necessarily joins N at some stage. In particular, at the time u_{j+i} was extracted from N,

$$\hat{\ell}_{u_{j+i}} = d_{G\left[\bar{V}_j \cup \{u_{j+i}\}\right]}(t_j, u_{j+i}) \leq d_{G\left[V_j'\right]}(t_j, v^j) + L(Q_j) \leq d_{G\left[V_j'\right]}(t_j, v^j)\,(1+2c_{\text{int}}\delta) \ ,$$

where the first equality follows by Claim 5.2, as $\hat{\ell}_{u_{j+i}}$ remains unchanged after extraction. We conclude that

$$\frac{\hat{\ell}_{u_{j+i}}}{D(u_{j+i})} \leq \frac{d_{G\left[V_j'\right]}(t_j, v^j)\,(1+2c_{\text{int}}\delta)}{D(v^j)\,(1-2c_{\text{int}}\delta)} \leq \frac{d_{G\left[V_j'\right]}(t_j, v^j)}{D(v^j)}\,(1+3\cdot 2c_{\text{int}}\delta) \leq (1+\delta)\,R_j \ .$$

We conclude that u_{j+i} joins V_j as required. ∎

In subsection 4.2 we defined charge function $f(\{x_Q\}_{Q\in\mathcal{Q}}) = \sum_{Q\in\mathcal{Q}} X(Q)\cdot L^+(Q)$, and in Lemma 4.2 we upper bounded its value (w.h.p.). In that analysis we exploit only Claim 4.1. Replacing it with Claim 5.5, the analysis still hold. That is, $\Pr[f(\{\tilde{X}(Q)\}_{Q\in\mathcal{Q}}) \geq 43\cdot d_G(t,t')] \leq k^{-3}$. Denote by $\mathcal{E}^{\text{fBig}}$ the event that for some pair of terminals t, t', $f(\tilde{X}(Q^1), \dots, \tilde{X}(Q^{\varphi})) \geq 43\cdot d_G(t,t')$. As previously, by union bound $\Pr\left[\mathcal{E}^{\text{fBig}}\right] < \frac{1}{2k}$. Denote by $\mathcal{E}^{\text{B}}$ the event that for some j, $R_j > c_d$. By Claim 4.5, $\Pr[\mathcal{E}^{\text{B}}] \leq \frac{1}{2k}$. We argue that assuming $\overline{\mathcal{E}^{\text{B}}}$ and $\overline{\mathcal{E}^{\text{fBig}}}$ (which happens with probability $1-\frac{1}{k}$), the distance between every pair of terminals t, t' in the minor returned by Algorithm 5.1 bounded by $O(\log k)\cdot d_G(v,u)$. This will conclude the proof of the distortion argument in Theorem 5.1. Recall that in contrast to Algorithm 3.1, the weight of the edge $\{t_i, t_j\}$ (if it exists) is $d_{G,V_i+V_j}(t_i,t_j)$ rather than $d_G(t_i,t_j)$; this will force some changes to our analysis. Recall the notation we used in Lemma 4.6: the path $P_{t,t'}$ is divided into consecutive detours $\mathcal{D}_{\ell_1}, \dots, \mathcal{D}_{\ell_{k'}}$. The leftmost (resp., rightmost) vertex in $\mathcal{D}_{\ell_j}$ is denoted by a_{ℓ_j} (resp., b_{ℓ_j}). Both a_{ℓ_j}, b_{ℓ_j} belong to V_{ℓ_j}, the cluster of t_{ℓ_j}. In particular, the graph G contains an edge between b_{ℓ_j} to $a_{\ell_{j+1}}$. Recall also that $t_{\ell_1} = t$ and $t_{\ell_k'} = t'$ (as each terminal covers itself). It holds that

$$
\begin{aligned}
d_M(t,t') &\le \sum_{j=1}^{k'-1} d_{G,V_{\ell_j}+V_{\ell_{j+1}}}(t_{\ell_j},t_{\ell_{j+1}}) \\
&\le \sum_{j=1}^{k'-1} \left[d_{G\left[V_{\ell_j}\right]}(t_{\ell_j},b_{\ell_j}) + d_G(b_{\ell_j},a_{\ell_{j+1}}) + d_{G\left[V_{\ell_{j+1}}\right]}(a_{\ell_{j+1}},t_{\ell_{j+1}})\right] \\
&\le c_d \cdot \sum_{j=1}^{k'-1} \left[d_G(t_{\ell_j},b_{\ell_j}) + d_G(b_{\ell_j},a_{\ell_{j+1}}) + d_G(a_{\ell_{j+1}},t_{\ell_{j+1}})\right] \\
&\le c_d \cdot \sum_{j=1}^{k'-1} \Big[d_G(t_{\ell_j},v^{\ell_j}) + d_G(v^{\ell_j},b_{\ell_j}) + d_G(b_{\ell_j},a_{\ell_{j+1}}) \\
&\qquad\qquad + d_G(a_{\ell_{j+1}},v^{\ell_{j+1}}) + d_G(v^{\ell_{j+1}},t_{\ell_{j+1}})\Big] \\
&\le c_d \cdot \left(\sum_{j=1}^{k'-1} d_G(v^{\ell_j},v^{\ell_{j+1}}) + 2\sum_{j=1}^{k'} d_G(t_{\ell_j},v^{\ell_j})\right) \\
&\le c_d \cdot \left(d_G(t,t') + 2c_d \cdot \sum_{j=1}^{k'} D(v^{\ell_j})\right) \\
&= O\left(\ln k\right) \cdot d_G(t,t')\ .
\end{aligned}
$$

The third inequality follows by our assumption $\overline{\mathcal{E}^{\mathrm{B}}}$, as for every index j and vertex $v \in V_j$, it holds that $d_{G[V_j]}(t_j,v) \le c_d \cdot D(v) \le c_d \cdot d_G(t_j,v)$. The fifth inequality follows as all $v^{\ell_j}, b_{\ell_j}, a_{\ell_{j+1}}, v^{\ell_{j+1}}$ lie on the same shortest path $P_{t,t'}$. The sixth inequality follows by $\overline{\mathcal{E}^{\mathrm{B}}}$ as $d_G(t_{\ell_j},v^{\ell_j}) \le d_{G\left[V_{\ell_j}\right]}(t_{\ell_j},v^{\ell_j}) \le c_d \cdot D(v^{\ell_j})$. The equality follows by inequality (4.6) and $\overline{\mathcal{E}^{\mathrm{fBig}}}$.

5.3. Runtime. For the implementation of Algorithm 5.1 and the `Fast-Create-Cluster` procedure we will use two basic data structures. The first one is a binary array to determine set membership of the vertices. It is folklore (see, for example, [1]) that an array could be initialized in constant time to be the all 0 array (that is, the empty set). Changing entry (that is, adding or deleting an element) also takes constant time. The second data structure is the Fibonacci heap (see [22]). Here each element has a key (some real number), and we can add a new element or decrease the value of the key in constant time. Finding the minimal element in the heap and deleting it takes $O(\log h)$ time (assuming there are currently h elements in the heap).

Before the execution of Algorithm 5.1, we compute the values $D(v)$ for all $v \in V$. This is done using an auxiliary graph G' where we add new vertex s with edges of weight 0 to all the terminals. Note that for every vertex v, the distance from s exactly equals $D(v)$. Thus we can simply run the Dijkstra algorithm from s to determine $D(v)$ for all $v \in V$. The runtime is $O(m + n\log n)$ (see [22]).

Next we give a detailed implementation of the `Fast-Create-Cluster` procedure. The sets V_j, U, and $V_\perp$ are stored using the arrays described above ($V_\perp$ will be a global variable). The set N will be stored using the Fibonacci heap, where the key value of $v \in N$ will be ℓ_v (i.e., $d_{G[V_j \cup \{v\}]}(v,t_j)$). Denote by $\mathcal{N}_j$ all the elements that belong to N at any stage of the execution of the `Fast-Create-Cluster` procedure (which created V_j). Let m_j denote the number of edges incident on vertices of V_j.

Each iteration of the while loop starts by deleting an element v with minimal key (of value $\hat{\ell}_v$) from N ($O(\log|\mathcal{N}_j|)$ time). Then we examine whether to add v to V_j (in $O(1)$ time). If v is rejected, we add v to U (in $O(1)$ time). Otherwise, v is added to V_j. In the latter case we go over each neighbor u of v. If $u \in U$ we do nothing. If $u \in N$, its key ℓ_u is updated to be $\min\{\ell_u, \ell_v + w(\{v,u\})\}$. Finally, if $u \in V_\perp \setminus (U \cup N)$, then u is added to N with the key $\ell_u \leftarrow \ell_v + w(\{v,u\})$. It is easy to verify that all the keys are indeed maintained with the correct values. Note that all this processing for u takes only $O(1)$ time. In particular, processing all neighbors throughout the `Fast-Create-Cluster` procedure takes $O(m_j)$ time. All the deletion of elements from the heap N takes $O(|\mathcal{N}_j|\log|\mathcal{N}_j|)$ time.

Next we bound the total cost of the k calls to the `Fast-Create-Cluster` procedure. $|\mathcal{N}_j|$ can be bounded from above by both m_j and n. Moreover, $\sum_j m_j \le 2m$, as every edge is incident on only two vertices. We provide two upper bounds on the running time:

$$O(n) + \sum_{j=1}^{k} O(m_j + |\mathcal{N}_j|\log|\mathcal{N}_j|) \le O\left(m + \sum_{j=1}^{k} m_j \log n\right) = O(m\log n)\ ,$$

$$O(n) + \sum_{j=1}^{k} O(m_j + |\mathcal{N}_j|\log|\mathcal{N}_j|) \le O\left(m + \sum_{j=1}^{k} n \log n\right) = O(m + nk\log n)\ .$$

Thus the total running time of these k calls is bounded by $O(m + \min\{m, nk\} \cdot \log n)$. Finally we bound the total runtime of Algorithm 5.1 without the calls to the `Create-Cluster`. It is straightforward that up line 9, where we create the minor M given the clusters, all computations took $O(n)$ time.[4] Using Claim 5.2, by the end of the for loop in Algorithm 5.1, for every j and $v \in V_j$ it holds that $\hat{\ell}_v = d_{G[V_j]}(t_j, v)$. In order to create the minor graph M, we go over all the edges iteratively, for every edge $\{v,u\} \in E$, such that $v \in V_j$, $u \in V_i$, and $i \ne j$. We add an edge $\{t_i, t_j\}$ to M (if it does not exist already). The weight of the edge updated to be the minimum between the current weight (∞ if it does not exist yet) and $\hat{\ell}_v + w(\{v,u\}) + \hat{\ell}_u$ (the keys at the time of extraction from N). It is straightforward that by the end of this procedure we will indeed compute the minor M, and each edge $\{t_i, t_j\}$ in M will have weight $d_{G,V_i+V_j}(t_i, t_j)$. This iterative process takes $O(m)$ time. Theorem 5.1 now follows.

6. Lower bounds on the performance of the algorithms. Chan et al. [9] gave a lower bound of 8 for the distortion in the SPR problem. This lower bound has not been improved since. This section is dedicated to lower bounding the performance of the various algorithms which were suggested for the problem. That is, while we do not provide better lower bounds for the SPR problem itself, we are able to lower bound the performance of the algorithms used so far.

In subsection 6.1 we prove that our analysis of the `Relaxed-Voronoi` algorithm (Algorithms 3.1 and 5.1) is asymptotically tight. That is, there is a graph family on which the achieved distortion is $\Theta(\log k)$. Next, in subsection 6.2, we provide a lower bound on the performance of the `Ball-growing` algorithm studied by [27, 11, 20]. Specifically, we provide (the same) graph family on which the `Ball-growing` algorithm incurs $\Omega(\sqrt{\log k})$ distortion. Recall that in [20], the author proved that the `Ball-growing` algorithm finds a minor with distortion $O(\log k)$. That is, while the

[4] In fact, the sampling of $g_1, \dots, g_k$ takes $O(k)$ time only w.h.p. But we will ignore this issue.

analysis of the `Ball-growing` algorithm still might be improved, it cannot be pushed further than $\Omega(\sqrt{\log k})$.

First, we show that the *expected* distortion incurred by the minor returned by the algorithms is large. Then, we deduce that with constant probability the (usual worst-case) distortion is also large. Formally, both algorithms are randomized and thus can be viewed as producing a distribution $\mathcal{D}$ over graph minors. Given such distribution $\mathcal{D}$, the expected distortion of the pair t, t' is $\mathbb{E}_{M\sim\mathcal{D}}\left[\frac{d_M(t,t')}{d_G(t,t')}\right]$. The overall expected distortion is the maximal expected distortion among all terminal pairs.

A final remark. Both algorithms used an arbitrary order over the terminals, in contrast to similar algorithms for other problems [8, 19] which consider a random order. Our lower bounds will still hold even if one replaces the arbitrary order with a random one.

6.1. Lower bound on the performance of the `Relaxed-Voronoi` algorithm. The following theorem provides a lower bound on the expected distortion incurred by Algorithm 3.1. The graphs which we will use for the lower bound are trees. As both Algorithm 3.1 and Algorithm 5.1 are identical where the input graph is a tree, the lower bound will also hold on Algorithm 5.1.

THEOREM 6.1. *Fix some $k \in \mathbb{N}$. There is a graph $G = (V, E, w)$ with terminal set K of size k such that the expected distortion of the minor returned by Algorithm* 3.1 *is* $\Omega(\log k)$.

Proof. We will assume that k is large enough, as otherwise $1 = \Omega(\log k)$ and hence every graph with k terminals provides a valid lower bound. Let G_k be the graph described in Figure 1.1 with parameter $\epsilon = 14\delta = \Theta(\frac{1}{\log k})$. Let X_j be an indicator for the event $v_j \in V_j$, that is, t_j covers v_j. For X_j to occur, it is enough that for every $i \neq j$, $d_G(t_i, v_j) > R_i \cdot D(v_j)$. That is, $R_i < 1 + |i-j| \cdot \epsilon$. By the definition of R_i,

$$\Pr\left[R_i \geq 1 + |i-j|\,\epsilon\right] = \Pr\left[g_i \geq \log_{1+\delta}\left(1 + |i-j|\,\epsilon\right)\right] = (1-p)^{\left\lceil \log_{1+\delta}(1+|i-j|\epsilon)-1\right\rceil} .$$

For i such that $|i-j| < \frac{1}{\epsilon}$, it holds that $\log_{1+\delta}\left(1 + |i-j|\,\epsilon\right) = \frac{\ln(1+|i-j|\epsilon)}{\ln(1+\delta)} \geq \frac{|i-j|\epsilon/2}{\delta}$, while for i such that $|i-j| \geq \frac{1}{\epsilon}$, $\log_{1+\delta}\left(1 + |i-j|\,\epsilon\right) \geq \frac{\ln 2}{\ln 1+\delta} \geq \frac{1}{2\delta}$. We conclude

$$\begin{aligned}
\Pr\left[X_i\right] &\geq \Pr\left[\forall_{j\neq i}\left(R_j < 1 + |i-j|\,\epsilon\right)\right] \\
&\geq 1 - \sum_{j\neq i} \Pr\left[R_j \geq 1 + |i-j|\,\epsilon\right] \\
&\geq 1 - 2\sum_{i=1}^{\left\lfloor \frac{1}{\epsilon}\right\rfloor}\left((1-p)^{\frac{i\epsilon/2}{\delta}-1}\right) - k\,(1-p)^{\frac{1}{2\delta}-1} .
\end{aligned}$$

Now, $\sum_{i=1}^{\left\lfloor \frac{1}{\epsilon}\right\rfloor}(1-p)^{\frac{i\epsilon/2}{\delta}} \leq \sum_{i=1}^{\infty}((1-p)^7)^i \leq \sum_{i=1}^{\infty}\frac{1}{4^i} = \frac{1}{4}\frac{1}{1-\frac{1}{4}} = \frac{1}{3}$, while $k\,(1-p)^{\frac{1}{2\delta}} = k\left(\frac{4}{5}\right)^{10\ln k} = k^{1-10\ln\frac{5}{4}} \leq \frac{1}{k}$. In particular $\Pr\left[X_i\right] \geq 1 - (1-p)^{-1} \cdot \left(2\cdot\frac{1}{3} + \frac{1}{k}\right) = \Omega(1)$.

Set $X = \sum_{i=2}^{k-1} X_i$. By linearity of expectation, $\mathbb{E}[X] = \Omega(k)$. Note that the distance from t_1 to t_k in the minor graph M_k equals $2 + (k-1)\,\epsilon + 2X$. We conclude

$$\mathbb{E}\left[\frac{d_{M_k}(t_1, t_m)}{d_{G_k}(t_1, t_m)}\right] = \frac{2 + (k-1)\,\epsilon + 2\mathbb{E}\left[X\right]}{2 + (k-1)\,\epsilon} = \frac{\Omega(k)}{O(k\epsilon)} = \Omega\left(\frac{1}{\epsilon}\right) = \Omega(\log k) .$$

□

COROLLARY 6.2. *Fix some $k \in \mathbb{N}$. There is a graph $G = (V, E, w)$ with terminal set K of size k such that with constant probability, the distortion incurred by the minor returned by Algorithm* 3.1 *is* $\Omega(\log k)$.

Proof. We will use the graph and notation from the proof of Theorem 6.1. Set $\mu = \mathbb{E}[\frac{d_{M_k}(t_1,t_m)}{d_{G_k}(t_1,t_m)}] = \Omega(\log k)$. Note the largest possible distortion is $\frac{2k-2+(k-1)\epsilon}{2+(k-1)\epsilon} = c\cdot\mu$ for some constant $c \geq 1$ (this distortion occurred exactly when each vertex v_j belongs to V_j). Denote by χ the event that $\frac{d_{M_k}(t_1,t_m)}{d_{G_k}(t_1,t_m)} \geq \frac{1}{2}\mu$. Then

$$\mu = \mathbb{E}\left[\frac{d_{M_k}(t_1,t_m)}{d_{G_k}(t_1,t_m)}\right] \leq \Pr[\chi] \cdot c\mu + (1 - \Pr[\chi]) \cdot \frac{1}{2}\mu\,,$$

therefore

$$\Pr[\chi] \geq \frac{1-\frac{1}{2}}{c-\frac{1}{2}} \geq \frac{1}{2c} = \Omega(1)\,.$$

Therefore, with constaint probability, the distortion is at least $\frac{1}{2}\mu = \Omega(\log k)$. ◻

6.2. Lower bound on the performance of the `Ball-Growing` algorithm. In this subsection we provide a lower bound on the performance of the `Ball-Growing` algorithm. For completeness, we give in Appendix B a full description of the `Ball-Growing` algorithm as it appeared in [20]. In particular, we will use the notation defined there. The `Ball-Growing` as described in [20] also had a modification step. As our lower bound example is a tree, this modification has no impact on the minor returned by the algorithm, and thus we can ignore it. Formally, a claim similar to Claim 3.3 can be proven.

THEOREM 6.3. *Fix some $k \in \mathbb{N}$. There is a graph $G = (V, E, w)$ with terminal set K of size k such that the expected distortion of the minor returned by the `Ball-Growing` algorithm is $\Omega(\sqrt{\log k})$.*

Proof. We will use the graph described in Figure 1.1 with modified parameters: the weight of an edge between terminal to Steiner vertex will be $2-\epsilon$, while the weight of an edge between two Steiner vertices will be 2ϵ for ϵ to be specified later. Note that the `Ball-Growing` algorithm assumes that the minimal distance between a terminal to a Steiner vertex in the input graph is exactly 1. In order to satisfy this condition we will add an additional Steiner vertex as a leaf connected to t_1 via an edge of unit weight. Note that this new vertex has no impact on the resulting minor whatsoever and therefore can be completely ignored.

As previously, we denote by X_j the indicator for the event $v_j \in V_j$. Following the analysis of Theorem 6.3, if we prove that $\Pr[X_j] = \Omega(1)$ (for arbitrary j) it will imply expected distortion of $\Omega(\frac{1}{\epsilon})$.

Let $\mathcal{R}_j$ be equal to R_j (the magnitude of t_j) at the end of the $m = \log_r 3 - 1$ round. For simplicity we will assume that m is an integer; otherwise the analysis will go through after slight modification of the parameters. Recall that $\mathcal{R}_j = \sum_{\ell=0}^{m} q_j^\ell$ where q_j^ℓ is distributed according to $\mathrm{Exp}(D \cdot r^\ell)$. Here $r = 1 + \frac{\delta}{\ln k}$, $\delta = \frac{1}{20}$, $D = \frac{\delta}{\ln k}$, and all the q_j^ℓ are independent. It holds that

$$\mathbb{E}\left[\mathcal{R}_j\right] = \sum_{\ell=0}^{m} D\cdot r^{\ell} = D\cdot\frac{r^{m+1}-1}{r-1} = 2\,,$$
$$\mathbb{V}\left[\mathcal{R}_j\right] = \mathbb{V}\left[\sum_{\ell=0}^{m} q_j^{\ell}\right] = \sum_{\ell=0}^{m}\mathbb{V}\left[q_j^{\ell}\right] = \sum_{\ell=0}^{m}\left(D\cdot r^{\ell}\right)^2$$
$$= D^2\cdot\frac{r^{2(m+1)}-1}{r^2-1} = \left(\frac{\delta}{\ln k}\right)^2\cdot\frac{9-1}{2\cdot\frac{\delta}{\ln k}+\left(\frac{\delta}{\ln k}\right)^2} \le 4\cdot\frac{\delta}{\ln k} = O\left(\frac{1}{\ln k}\right)\ ,$$

where we used linearity of expectation and independence. In order that X_j will occur, it is enough that $\mathcal{R}_j \ge d(t_j, v_j)$, while for every $j' \ne j$, $\mathcal{R}_j < d(t_{j'}, v_j)$. Using the Chebyshev inequality,

$$\Pr\left[\mathcal{R}_j \ge d(t_j, v_j)\right] = \Pr\left[\mathcal{R}_j \ge 2-\epsilon\right] \ge \Pr\left[\left|\mathcal{R}_j - \mathbb{E}\left[\mathcal{R}_j\right]\right| < \epsilon\right] \ge 1 - \frac{\mathbb{V}\left[\mathcal{R}\right]}{\epsilon^2}\ ,$$
$$\Pr\left[\mathcal{R}_{j'} \ge d(t_{j'}, v_j)\right] \le \Pr\left[\left|\mathcal{R}_{j'} - \mathbb{E}\left[\mathcal{R}_{j'}\right]\right| \ge \left(2\left|j-j'\right|-1\right)\epsilon\right] \le \frac{\mathbb{V}\left[\mathcal{R}\right]}{\left(2\left|j-j'\right|-1\right)^2\cdot\epsilon^2}\ .$$

By the union bound, the probability that for some $j' \ne j$, $\mathcal{R}_{j'} \ge d(t_{j'}, v_j)$ is bounded by

$$\sum_{j\ne j'}\Pr\left[\mathcal{R}_{j'} \ge d(t_{j'}, v_j)\right] < \frac{\mathbb{V}\left[\mathcal{R}\right]}{\epsilon^2}\cdot 2\cdot\sum_{i=1}^{\infty}\frac{1}{i^2} = \frac{\mathbb{V}\left[\mathcal{R}\right]}{\epsilon^2}\cdot\frac{\pi^2}{3}\ .$$

We conclude

$$\Pr\left[X_j\right] \ge \Pr\left[\mathcal{R}_{j'} \ge d(t_{j'}, v_j)\right]\cdot\left(1-\sum_{j\ne j'}\Pr\left[\mathcal{R}_{j'} \ge d(t_{j'}, v_j)\right]\right)$$
$$\ge \left(1-\frac{\mathbb{V}\left[\mathcal{R}\right]}{\epsilon^2}\right)\left(1-\frac{\mathbb{V}\left[\mathcal{R}\right]}{\epsilon^2}\cdot\frac{\pi^2}{3}\right) = 1 - O\left(\frac{1}{\epsilon^2\ln k}\right) = \Omega(1)$$

for $\epsilon = \Theta(\frac{1}{\sqrt{\log k}})$. The theorem now follows. ∎

Following the lines of the proof of Corollary 6.2, we conclude as follows.

COROLLARY 6.4. *Fix some $k \in \mathbb{N}$. There is a graph $G = (V, E, w)$ with terminal set K of size k such that with constant probability, the distortion of the minor returned by the* `Ball-Growing` *algorithm is* $\Omega(\sqrt{\log k})$

Remark 6.5. Theorem 6.3 can also be proved using concentration bounds. However, the lower bound remains $\Omega(\sqrt{\log k})$ so we provided the more basic proof using the Chebyshev inequality. Nevertheless, the curious reader can find the required concentration bounds for such a proof in Appendix A.

7. Discussion. In this paper we proved an $O(\log k)$ upper bound for the SPR problem, improving the previous $O(\log^2 k)$ upper bound by [11]. The lower bound is still only 8 [9]. Closing this gap remains an intriguing open problem. Both the `Relaxed-Voronoi` and `Ball-growing` algorithms proceed by creating random terminal partitions. These partitions are determined using random parameters, which are chosen with no consideration whatsoever of the input graph G. In contrast, the optimal tree algorithm of [24] is a deterministic recursive algorithm which make decisions after considering the tree structure at hand. It seems that the input-oblivious approach of the `Relaxed-Voronoi` and `Ball-growing` algorithms is doomed for failure, and in fact, both these algorithms already fail to achieve constant distortion on a simple tree example. As a conclusion, input-sensitive approaches seem to be more promising for future attempts to resolve the SPR problem.

In a follow-up paper with Krauthgamer and Trabelsi [21], we used the `Relaxed-Voronoi` algorithm in order to re-prove Gupta's [24] upper bound of 8. Formally, let $r \in V$ be an arbitrary vertex and order the terminals w.r.t. their distances from r (that is, $d(t_1, r) \le d(t_2, r) \le \dots d(t_k, r)$). Surprisingly, given a tree, if we run the `Relaxed-Voronoi` algorithm w.r.t. the order specified above (instead of an arbitrary order), and all magnitudes R_j are exactly 3, we will get a tree minor with distortion at most 8. This example demonstrates that one can use the `Relaxed-Voronoi` algorithm also in an input-sensitive manner in order to achieve optimal results.

We would like to emphasize two additional open problems:

- Expected distortion: Currently the state of the art for usual (worst-case) distortion and expected distortion for the SPR problem is the same. Both have $O(\log k)$ upper bound and $\Omega(1)$ lower bound. There are cases where much better results can be achieved for expected distortion (e.g., embedding a graph into a tree must incur distortion $\Omega(n)$, while a distribution over embeddings into trees can have expected distortion $O(\log n)$ [19]). What are the right bounds for expected distortion in the SPR problem?
- Special graph families: Basu and Gupta [5] showed that constant distortion for the SPR problem can be achieved on outer-planar graphs. It will be very interesting to achieve better upper bounds for planar graphs, and more generally for minor-free graphs, bounded treewidth graphs, etc. In the expected distortion regime, an $O(1)$ upper bound is already known [17] for minor-free graphs. Possibly one can use the `Relaxed-Voronoi` algorithm with a clever choice of order and magnitudes in order to achieve such results.

Appendix A. Concentration bounds for sum of exponential distributions.

LEMMA A.1. *Suppose $X_1, \dots, X_n$'s are independent random variables, where each X_i is distributed according to $\mathsf{Exp}(\lambda_i)$. Let $X = \sum_i X_i$ and $\lambda_M = \max_i \lambda_i$. Set $\mu = \mathbb{E}[X] = \sum_i \lambda_i$.*
For $0 < t \le \frac{1}{2\lambda_M}$, and $\alpha \ge 2t\lambda_M$,

$$\Pr[X \ge (1+\alpha)\mu] \le \exp\left(-t\mu \cdot (\alpha - 2t\lambda_M)\right),$$
$$\Pr[X \le (1-\alpha)\mu] \le \exp\left(-t\mu\left(\alpha - t\lambda_M\right)\right) \ .$$

Proof. For each X_i, the moment generating function w.r.t. t equals

$$\mathbb{E}\left[e^{tX_i}\right] = \frac{1}{1-t\lambda_i} = 1 + t\lambda_i\left(\sum_{\ell \ge 0} (t\lambda_i)^{\ell}\right) \le 1 + t\lambda_i\left(1 + 2t\lambda_i\right) \le e^{t\lambda_i(1+2t\lambda_i)}.$$

Using the Markov inequality,

$$\begin{aligned}
\Pr\left[X \geq (1+\alpha)\mu\right] &= \Pr\left[e^{tX} \geq e^{t(1+\alpha)\mu}\right] \\
&\leq \mathbb{E}\left[e^{tX}\right] \cdot e^{-t(1+\alpha)\mu} \\
&= e^{-t(1+\alpha)\sum_\ell \lambda_\ell} \cdot \prod_\ell \mathbb{E}\left[e^{tX_\ell}\right] \\
&\leq e^{-(1+\alpha)\sum_\ell t\lambda_\ell} \cdot e^{\sum_\ell t\lambda_\ell(1+2t\lambda_\ell)} \\
&= e^{\sum_\ell (t\lambda_\ell \cdot (2t\lambda_\ell - \alpha))} \\
&\leq e^{\left(\sum_\ell t\lambda_\ell\right)\cdot(2t\lambda_M - \alpha)} = e^{-t\mu\cdot(\alpha - 2t\lambda_M)},
\end{aligned}$$

where in the second equality we use the fact that $\{X_i\}_i$ are independent.

For the second inequality, it holds that

$$\mathbb{E}\left[e^{-tX_i}\right] = \frac{1}{1+t\lambda_i} = \sum_{\ell\geq 0} (-1)^\ell (t\lambda_i)^\ell \leq 1 - t\lambda_i(1-t\lambda_i) \leq e^{-t\lambda_i(1-t\lambda_i)} .$$

Therefore,

$$\begin{aligned}
\Pr\left[X \leq (1-\alpha)\mu\right] &= \Pr\left[e^{-tX} \geq e^{-t(1-\alpha)\mu}\right] \\
&\leq \mathbb{E}\left[e^{-tX}\right] / e^{-t(1-\alpha)\mu} \\
&= e^{t(1-\alpha)\mu} \cdot \Pi_\ell \mathbb{E}\left[e^{-tX_\ell}\right] \\
&\leq e^{(1-\alpha)\sum_\ell t\lambda_\ell} \cdot e^{-\sum_\ell t\lambda_\ell(1-t\lambda_\ell)} \\
&= e^{-\sum_\ell t\lambda_\ell(\alpha - t\lambda_\ell)} \\
&\leq e^{-t\mu(\alpha - t\lambda_M)} .
\end{aligned}$$

∎

We derive the following corollary.

COROLLARY A.2. *Suppose $X_1, \ldots, X_n$ are independent random variables, where $X_i \sim \mathsf{Exp}(\lambda_i)$. Let $X = \sum_i X_i$ and $\lambda_M = \max_i \lambda_i$. Set $\mu = \mathbb{E}[X] = \sum_i \lambda_i$. Then,*

$$\begin{aligned}
&\textit{For } \alpha \leq 2: \ \Pr\left[X \geq (1+\alpha)\mu\right] \leq \exp\left(-\frac{\alpha^2\mu}{8\lambda_M}\right) , \\
&\textit{For } \alpha \leq 1: \ \Pr\left[X \leq (1-\alpha)\mu\right] \leq \exp\left(-\frac{\alpha^2\mu}{4\lambda_M}\right) .
\end{aligned}$$

For the first inequality we choose the parameter $t = \frac{\alpha}{2} \cdot \frac{1}{2\lambda_M}$, while for the second inequality we choose the parameter $t = \alpha \cdot \frac{1}{2\lambda_M}$.

Appendix B. The `Ball-Growing` algorithm. The `Ball-Growing` algorithm assumes w.l.o.g. that the minimal distance between a terminal to a Steiner vertex in the input graph is exactly 1. Throughout the execution of the algorithm each terminal t_j is associated with a radius R_j and cluster $V_j \subset V$. The algorithm iteratively grows clusters $V_1, \ldots, V_k$ around the terminals. Once some vertex v joins some cluster V_j, it will stay there. When all the vertices are clustered, the algorithm terminates. Initially the cluster V_j contains only the terminal t_j, while R_j equals 0. The algorithm will have rounds, where each round consist of k steps. In step j of round ℓ, the algorithm samples a number q_j^ℓ according to distribution $\mathsf{Exp}(D \cdot r^\ell)$ (note that the mean of the distribution grows by a factor of r in each round). The radius R_j grows by q_j^ℓ. We consider the graph induced by the unclustered vertices $V_\perp$ union V_j. Every unclustered vertex of distance at most R_j from t_j in $G[V_\perp \cup V_j]$ joins V_j.

Algorithm B.1. $M = \texttt{Ball-Growing}(G = (V,E), w, K = \{t_1, \ldots, t_k\})$.

```
Set r ← 1 + δ/ln k, where δ = 1/20.
Set D ← δ / ln k.
For each j ∈ [k], set V_j ← {t_j}, and set R_j ← 0.
Set V_⊥ ← V \ (∪_{j=1}^k V_j).
Set ℓ ← 0.
while (∪_{j=1}^k V_j) ≠ V do
  for j from 1 to k do
    Choose independently at random q_j^ℓ distributed according to Exp(D · r^ℓ).
    Set R_j ← R_j + q_j^ℓ.
    Set V_j ← B_{G[V_⊥ ∪ V_j]}(t_j, R_j). .        // This is the same as
    V_j ← V_j ∪ B_{G[V_⊥ ∪ V_j]}(t_j, R_j).
    Set V_⊥ ← V \ (∪_{j=1}^k V_j).
  end for
  ℓ ← ℓ + 1.
end while
return the terminal-centered minor M of G induced by V_1, ..., V_k.
```

Appendix C. Index.

Preliminaries.

d_G: shortest path metric in G.
$G[A]$: graph induced by A.
$K = \{t_1, \ldots, t_k\}$: set of terminals.
$D(v) = \min_{t \in K} d_G(v,t)$.
Terminal partition: partition $\{V_1, \ldots, V_k\}$ of V, s.t. for every i, $t_i \in V_i$ and V_i is connected.
Induced minor: given terminal partition $\{V_1, \ldots, V_k\}$, the induced minor obtained by contracting each V_i into the super vertex t_i. The weight of the edge $\{t_i, t_j\}$ (if it exists) set to be $d_G(t_i, t_j)$.
Distortion of induced minor: $\max_{i,j} \frac{d_M(t_i,t_j)}{d_G(t_i,t_j)}$.
$\mathsf{Geo}(p)$: geometric distribution with parameter λ.
$\mathsf{Exp}(\lambda)$: exponential distribution with parameter p.

Modification.
Every edge on $P_{t,t'}$ has weight at most $c_w \cdot d_G(t,t')$.

Constants.

$p = \frac{1}{5}$: parameter of the geometric distribution.
$\delta = \frac{1}{20 \cdot \ln k}$: jumps in R_j are of magnitude $1+\delta$.
$c_w = \frac{\delta}{24}$.
$c_{\mathbf{int}} = \frac{1}{6}$: governs the size of interval in the partition $\mathcal{Q}$ of $P_{t,t'}$.
$c_{\mathbf{con}} = \frac{1}{2}$: used to bound the variation of the charge function from its expectation.
$c_d = e^2$: bound on the maximal size of R_j.

Events.

$\mathcal{E}^{\mathrm{fBig}}$: denotes that for some pair of terminals t, t', $f(\{X(Q)\}_{Q \in \mathcal{Q}} \geq 43 \cdot d_G(t,t')$.
$\mathcal{E}^{\mathrm{B}}$: denotes that there exist j such that $R_j > c_d$.

Notation.

V_j: cluster of t_j.
R_j: magnitude of the cluster of t_j.
$V_\perp$: set of unclustered (uncovered) vertices.
$P_{t,t'} = \{t = v_0, \ldots, v_\gamma = t'\}$: shortest path from t to t'.
$L(\{v_a, v_{a+1}, \ldots, v_b\}) = d_G(v_a, v_b)$: internal length.
$L^+(\{v_a, v_{a+1}, \ldots, v_b\}) = d_G(v_{a-1}, v_{b+1})$: external length.
$\mathcal{Q}$: partition of $P_{t,t'}$ into intervals Q.
a_j: the leftmost active vertex covered by t_j.
b_j: the rightmost active vertex covered by t_j.
$\mathcal{D}_j = \{a_j, \ldots, b_j\}$: detour created by terminal t_j.
Slice maximal subinterval (of some Q) of active vertices.
r_v: minimal choice of R_j such that v joins V_j.
v^j: vertex with the minimal r_v (among active vertices).
Q_j: interval containing v_j.
S_j: slice containing v_j.
$f(\{x_Q\}_{Q \in \mathcal{Q}})$: $= \sum_{Q \in \mathcal{Q}} x_Q \cdot L^+(Q)$, charge function.
B_Q: a coin box which resembles the interval Q.
$d_{G,V_i+V_j}(t_i,t_j)$: the weight of the shortest path in G between t_1 and t_2 that uses only vertices from $V_i \cup V_j$ and only a single crossing edge between V_i to V_j.

Counters.

$\mathcal{S}(Q)$: (current) number of slices in interval Q.
$X(Q)$: number of detours the interval Q is (currently) charged for.
$\tilde{X}(Q)$: number of detours the interval Q is charged for by the end of Algorithm 3.1.
$Z(Q)$: number of active coins in B_Q. Each coin is active when added to the box.
$Y(Q)$: number of inactive coins in B_Q. A coin becomes inactive after tossing.
$\tilde{Y}(Q)$: number of inactive coins in B_Q by the end of the process.

Acknowledgment. The author would like to thank his advisors Ofer Neiman, for fruitful discussions, and Robert Krauthgamer, for useful comments.

REFERENCES

[1] A. V. Aho and J. E. Hopcroft, *The Design and Analysis of Computer Algorithms*, Addison-Wesley, Boston, MA, 1974.

[2] A. Andoni, A. Gupta, and R. Krauthgamer, *Towards* $(1+\epsilon)$*-approximate flow sparsifiers*, in Proceedings of the 25th Annual ACM-SIAM Symposium on Discrete Algorithms, Portland, OR, 2014, pp. 279–293, https://doi.org/10.1137/1.9781611973402.20.

[3] Y. Bartal, *Probabilistic approximations of metric spaces and its algorithmic applications*, in Proceedings of the 37th Annual Symposium on Foundations of Computer Science, Burlington, VT, 1996, pp. 184–193, https://doi.org/10.1109/SFCS.1996.548477.

[4] Y. Bartal, A. Filtser, and O. Neiman, *On notions of distortion and an almost minimum spanning tree with constant average distortion*, in Proceedings of the 27th Annual ACM-SIAM Symposium on Discrete Algorithms, Arlington, VA, 2016, pp. 873–882, https://doi.org/10.1137/1.9781611974331.ch62.

[5] A. Basu and A. Gupta, *Steiner Point Removal in Graph Metrics*, manuscript, http://www.math.ucdavis.edu/~abasu/papers/SPR.pdf (2008).

[6] J. D. Batson, D. A. Spielman, and N. Srivastava, *Twice-Ramanujan sparsifiers*, SIAM J. Comput., 41 (2012), pp. 1704–1721, https://doi.org/10.1137/090772873.

[7] A. A. Benczúr and D. R. Karger, *Approximating* s-t *minimum cuts in* $\tilde{O}(n^2)$ *time*, in Proceedings of the 28th Annual ACM Symposium on the Theory of Computing, Philadelphia, PA, 1996, pp. 47–55, https://doi.org/10.1145/237814.237827.

[8] G. Călinescu, H. J. Karloff, and Y. Rabani, *Approximation algorithms for the* 0*-extension problem*, SIAM J. Comput., 34 (2004), pp. 358–372.

[9] T.-H. Chan, D. Xia, G. Konjevod, and A. Richa, *A tight lower bound for the Steiner point removal problem on trees*, in Proceedings of the 9th International Conference on Approximation Algorithms for Combinatorial Optimization Problems, and 10th International Conference on Randomization and Computation, Springer-Verlag, Berlin, 2006, pp. 70–81, https://doi.org/10.1007/11830924_9.

[10] M. Charikar, T. Leighton, S. Li, and A. Moitra, *Vertex sparsifiers and abstract rounding algorithms*, in Proceedings of the 51th Annual IEEE Symposium on Foundations of Computer Science, Las Vegas, NV, 2010, pp. 265–274, https://doi.org/10.1109/FOCS.2010.32.

[11] Y. K. Cheung, *Steiner point removal—distant terminals don't (really) bother*, in Proceedings of the 29th Annual ACM-SIAM Symposium on Discrete Algorithms, New Orleans, LA, 2018, pp. 1353–1360.

[12] Y. K. Cheung, G. Goranci, and M. Henzinger, *Graph minors for preserving terminal distances approximately—lower and upper bounds*, in 43rd International Colloquium on Automata, Languages, and Programming, Rome, Italy, 2016, pp. 131:1–131:14, https://doi.org/10.4230/LIPIcs.ICALP.2016.131.

[13] J. Chuzhoy, *On vertex sparsifiers with Steiner nodes*, in Proceedings of the 44th Symposium on Theory of Computing Conference, New York, 2012, pp. 673–688, https://doi.org/10.1145/2213977.2214039.

[14] D. Coppersmith and M. Elkin, *Sparse source-wise and pair-wise distance preservers*, in Proceedings of the 16th Annual ACM-SIAM Symposium on Discrete Algorithms, SIAM, Philadelphia, PA, 2005, pp. 660–669, http://dl.acm.org/citation.cfm?id=1070432.1070524.

[15] M. Elkin, A. Filtser, and O. Neiman, *Prioritized metric structures and embedding*, in Proceedings of the 47th Annual ACM on Symposium on Theory of Computing, Portland, OR, 2015, pp. 489–498, https://doi.org/10.1145/2746539.2746623.

[16] M. Elkin, A. Filtser, and O. Neiman, *Terminal embeddings*, Theoret. Comput. Sci., 697 (2017), pp. 1–36, https://doi.org/10.1016/j.tcs.2017.06.021.

[17] M. Englert, A. Gupta, R. Krauthgamer, H. Räcke, I. Talgam-Cohen, and K. Talwar, *Vertex sparsifiers: New results from old techniques*, SIAM J. Comput., 43 (2014), pp. 1239–1262, https://doi.org/10.1137/130908440.

[18] J. Fakcharoenphol, C. Harrelson, S. Rao, and K. Talwar, *An improved approximation algorithm for the* 0*-extension problem*, in Proceedings of the 14th Annual ACM-SIAM Symposium on Discrete Algorithms, Baltimore, MD, 2003, pp. 257–265, http://dl.acm.org/citation.cfm?id=644108.644153.

[19] J. Fakcharoenphol, S. Rao, and K. Talwar, *A tight bound on approximating arbitrary metrics by tree metrics*, J. Comput. System Sci., 69 (2004), pp. 485–497, https://doi.org/10.1016/j.jcss.2004.04.011.

[20] A. Filtser, *Steiner point removal with distortion* O*(log* k*)*, in Proceedings of the 29th Annual ACM-SIAM Symposium on Discrete Algorithms, New Orleans, LA, 2018, pp. 1361–1373, https://doi.org/10.1137/1.9781611975031.90.

[21] A. Filtser, R. Krauthgamer, and O. Trabelsi, *Relaxed voronoi: A simple framework for terminal-clustering problems*, in Proceedings of the 2nd Symposium on Simplicity in Algorithms, San Diego, CA, 2019, pp. 10:1–10:14, https://doi.org/10.4230/OASIcs.SOSA.2019.10.

[22] M. L. Fredman and R. E. Tarjan, *Fibonacci heaps and their uses in improved network optimization algorithms*, J. ACM, 34 (1987), pp. 596–615, https://doi.org/10.1145/28869.28874.

[23] G. Goranci, M. Henzinger, and P. Peng, *Improved guarantees for vertex sparsification in planar graphs*, in Proceedings of the 25th Annual European Symposium on Algorithms, Vienna, Austria, 2017, pp. 44:1–44:14, https://doi.org/10.4230/LIPIcs.ESA.2017.44.

[24] A. Gupta, *Steiner points in tree metrics don't (really) help*, in Proceedings of the 12th Annual ACM-SIAM Symposium on Discrete Algorithms, Philadelphia, PA, 2001, pp. 220–227, http://dl.acm.org/citation.cfm?id=365411.365448.

[25] A. Gupta, V. Nagarajan, and R. Ravi, *An improved approximation algorithm for requirement cut*, Oper. Res. Lett., 38 (2010), pp. 322–325.

[26] L. Kamma, R. Krauthgamer, and H. L. Nguyen, *Cutting corners cheaply, or how to remove Steiner points*, in Proceedings of SODA, 2014, pp. 1029–1040.

[27] L. Kamma, R. Krauthgamer, and H. L. Nguyen, *Cutting corners cheaply, or how to remove Steiner points*, SIAM J. Comput., 44 (2015), pp. 975–995, https://doi.org/10.1137/140951382.

[28] T. Kavitha and N. M. Varma, *Small stretch pairwise spanners*, in Automata, Languages, and Programming Part I, Lecture Notes in Comput. Sci. 7965, Springer-Verlag, Berlin, 2013, pp. 601–612, https://doi.org/10.1007/978-3-642-39206-1_51.

[29] R. Krauthgamer, H. L. Nguyen, and T. Zondiner, *Preserving terminal distances using minors*, SIAM J. Discrete Math., 28 (2014), pp. 127–141, https://doi.org/10.1137/120888843.

[30] R. Krauthgamer and I. Rika, *Mimicking networks and succinct representations of terminal cuts*, in Proceedings of the 24th Annual ACM-SIAM Symposium on Discrete Algorithms, New Orleans, LA, 2013, pp. 1789–1799, https://doi.org/10.1137/1.9781611973105.128.

[31] R. Krauthgamer and I. Rika, *Refined Vertex Sparsifiers of Planar Graphs*, CoRR abs/1702.05951, 2017.

[32] F. T. Leighton and A. Moitra, *Extensions and limits to vertex sparsification*, in Proceedings of the 42nd ACM Symposium on Theory of Computing, Cambridge, MA, 2010, pp. 47–56, https://doi.org/10.1145/1806689.1806698.

[33] N. Linial and M. E. Saks, *Decomposing graphs into regions of small diameter*, in Proceedings of the 2nd Annual ACM/SIGACT-SIAM Symposium on Discrete Algorithms, San Francisco, CA, 1991, pp. 320–330, http://dl.acm.org/citation.cfm?id=127787.127848.

[34] K. Makarychev and Y. Makarychev, *Metric extension operators, vertex sparsifiers and Lipschitz extendability*, in Proceedings of the 51th Annual IEEE Symposium on Foundations of Computer Science, Las Vegas, NV, 2010, pp. 255–264, https://doi.org/10.1109/FOCS.2010.31.

[35] G. L. Miller, R. Peng, A. Vladu, and S. C. Xu, *Improved parallel algorithms for spanners and hopsets*, in Proceedings of the 27th ACM on Symposium on Parallelism in Algorithms and Architectures, Portland, OR, 2015, pp. 192–201, https://doi.org/10.1145/2755573.2755574.

[36] A. Moitra, *Approximation algorithms for multicommodity-type problems with guarantees independent of the graph size*, in Proceedings of the 50th Annual IEEE Symposium on Foundations of Computer Science, Atlanta, GA, 2009, pp. 3–12, https://doi.org/10.1109/FOCS.2009.28.

[37] D. Peleg and A. A. Schäffer, *Graph spanners*, J. Graph Theory, 13 (1989), pp. 99–116, https://doi.org/10.1002/jgt.3190130114.

[38] L. Roditty, M. Thorup, and U. Zwick, *Deterministic constructions of approximate distance oracles and spanners*, in Automata, Languages and Programming, Lecture Notes in Comput. Sci. 3580, Springer-Verlag, Berlin, 2005, pp. 261–272, https://doi.org/10.1007/11523468_22.

[39] M. Thorup and U. Zwick, *Approximate distance oracles*, J. ACM, 52 (2005), pp. 1–24, https://doi.org/10.1145/1044731.1044732.

Part V

Sparsification of Two-Variable Valued CSPs

SIAM J. DISCRETE MATH.
Vol. 31, No. 2, pp. 1263–1276

SPARSIFICATION OF TWO-VARIABLE VALUED CONSTRAINT SATISFACTION PROBLEMS*

ARNOLD FILTSER† AND ROBERT KRAUTHGAMER‡

Abstract. A valued constraint satisfaction problem (VCSP) instance (V, Π, w) is a set of variables V with a set of constraints Π weighted by w. Given a VCSP instance, we are interested in a reweighted subinstance $(V, \Pi' \subset \Pi, w')$ that preserves the value of the given instance (under every assignment to the variables) within factor $1 \pm \epsilon$. A well-studied special case is cut sparsification in graphs, which has found various applications. We show that a VCSP instance consisting of a single boolean predicate $P(x, y)$ (e.g., for cut, $P = \mathsf{XOR}$) can be sparsified into $O(|V|/\epsilon^2)$ constraints iff the number of inputs that satisfy P is anything but one (i.e., $|P^{-1}(1)| \neq 1$). Furthermore, this sparsity bound is tight unless P is a relatively trivial predicate. We conclude that also systems of 2SAT (or 2LIN) constraints can be sparsified.

Key words. valued constraint satisfaction problem, cut sparsification, boolean predicates, MAX-CSP

AMS subject classifications. 68Q25, 68W25

DOI. 10.1137/15M1046186

1. Introduction.

The seminal work of Benczúr and Karger [4] showed that every edge-weighted undirected graph $G = (V, E, w)$ admits cut sparsification within factor $(1+\epsilon)$ using $O(\epsilon^{-2} n \log n)$ edges, where we denote throughout $n = |V|$. To state it more precisely, assume that edge weights are always non negative and let $\mathsf{Cut}_G(S)$ denote the total weight of edges in G that have exactly one endpoint in S. Then for every such G and $\epsilon \in (0, 1)$, there is a reweighted subgraph $G_\epsilon = (V, E_\epsilon \subseteq E, w_\epsilon)$ with $|E_\epsilon| \leq O(\epsilon^{-2} n \log n)$ edges such that

$$\forall S \subset V, \qquad \mathsf{Cut}_{G_\epsilon}(S) \in (1 \pm \epsilon) \cdot \mathsf{Cut}_G(S), \tag{1}$$

and moreover, such G_ϵ can be computed efficiently.

This sparsification methodology turned out to be very influential. The original motivation was to speed up algorithms for cut problems—one can compute a cut sparsifier of the input graph and then solve an optimization problem on the sparsifier—and indeed this has been a tremendously effective approach; see, e.g., [4, 5, 10, 14, 12]. Another application of this remarkable notion is to reduce space requirements, either when storing the graph or in streaming algorithms [1]. In fact, followup work offered several refinements, improvements, and extensions (such as to spectral sparsification or to cuts in hypergraphs, which in turn have more applications); see, e.g., [16, 17, 15, 7, 8, 9, 13, 3, 11]. The current bound for cut sparsification is $O(n/\epsilon^2)$ edges, proved by Batson, Spielman, and Srivastava [3], and it is known to be tight [2].

We study the analogous problem of sparsifying constraint satisfaction problems (CSPs), which was raised in [11, section 4] and goes as follows. Given a set of

*Received by the editors October 30, 2015; accepted for publication (in revised form) January 27, 2017; published electronically June 22, 2017. A preliminary version is available at arXiv:1509.01844.
http://www.siam.org/journals/sidma/31-2/M104618.html

Funding: The first author was partially supported by the Lynn and William Frankel Center for Computer Sciences. The second author's work was supported in part by Israel Science Foundation grant 897/13 and US-Israel BSF grant 2010418.

†Ben-Gurion University of the Negev, Beer-Sheva 8410501, Israel (arnoldf@cs.bgu.ac.il).
‡Weizmann Institute of Science, Rehovot 76100, Israel (robert.krauthgamer@weizmann.ac.il).

constraints on n variables, the goal is to construct a sparse subinstance that has approximately the same value as the original instance under *every possible assignment*; see section 2 for a formal definition. Such sparsification of CSPs can be used to reduce storage space and running time of many algorithms.

We restrict our attention to two-variable constraints (i.e., of arity 2) over boolean domain (i.e., alphabet of size 2). To simplify matters even further we shall start with the case where all the constraints use the same predicate $P : \{0,1\}^2 \to \{0,1\}$. This restricted case of CSP sparsification already generalizes cut sparsification—simply represent every vertex $v \in V$ by a variable x_v and every edge $(v,u) \in E$ by the constraint $x_v \neq x_u$.

Observe that such CSPs also capture other interesting graph problems, such as the *uncut edges* (using the predicate $x_v = x_u$), *covered edges* (using the predicate $x_v \vee x_u$), or *directed-cut edges* (using the predicate $x_v \wedge \neg x_u$). Even though these graph problems are well-known and extensively studied, we are not aware of any sparsification results for them, and at a first glance such sparsification may even seem surprising, because these problems do not have the combinatorial structure exploited by [4] (a bound on the number of approximately minimum cuts) or the linear-algebraic description used by [15, 3] (as quadratic forms over Laplacian matrices).

Results. For CSPs consisting of a single predicate $P : \{0,1\}^2 \to \{0,1\}$, we show in Theorem 3.7 that a $(1+\epsilon)$-sparsifier of size $O(n/\epsilon^2)$ always exists iff $|P^{-1}(1)| \neq 1$ (i.e., P has 0, 2, 3, or 4 satisfying inputs). Observe that the latter condition includes the two graphical examples above uncut edges and covered edges but excludes directed-cut edges. We further show in Theorem 4.1 that our sparsity bound above is tight, except for some relatively trivial predicates P. We then build on our sparsification result in section 5 to obtain $(1+\epsilon)$-sparsifiers for other CSPs, including 2SAT (which uses four predicate types) and 2LIN (which uses two predicate types).

Finally, we explore future directions, such as more general predicates and a generalization of the sparsification paradigm to sketching schemes. In particular, we see that the above dichotomy according to number of satisfying inputs to the predicate extends to sketching.

2. Two-variable boolean predicates and digraphs. A *predicate* is a function $\mathsf{P} : \{0,1\}^2 \to \{0,1\}$ (recall we restrict ourselves throughout to two variables and a boolean domain). Given a set of variables V, a *constraint* $\langle (v,u), \mathsf{P}\rangle$ consists of a predicate P and an ordered pair (v,u) of variables from V. For an assignment $A : V \to \{0,1\}$, we say that A *satisfies* the constraint whenever $\mathsf{P}(A(v), A(u)) = 1$. A valued constraint satisfaction problem (VCSP) instance $\mathcal{I}$ is a triple (V, Π, w), where V is a set of variables, Π is a set of constraints over V (each of the form $\pi_i = \langle (v_i, u_i), p_i\rangle$), and $w : \Pi \to \mathbb{R}_+$ is a weight function. The *value* of an assignment $A : V \to \{0,1\}$ is the total weight of the satisfied constraints, i.e.,

$$\mathrm{Val}_{\mathcal{I}}(A) := \sum_{\pi_i \in \Pi} w(\pi_i) \cdot p_i(A(v_i), A(u_i)).$$

For $\epsilon \in (0,1)$, an *ϵ-sparsifier* of $\mathcal{I}$ is a (reweighted) subinstance $\mathcal{I}_\epsilon = (V, \Pi_\epsilon \subseteq \Pi, w_\epsilon)$ where

$$\forall A : V \to \{0,1\}, \qquad \mathrm{Val}_{\mathcal{I}_\epsilon}(A) \in (1 \pm \epsilon) \cdot \mathrm{Val}_{\mathcal{I}}(A).$$

The goal is to minimize the number of constraints, i.e., $|\Pi_\epsilon|$. There are 16 different predicates $\mathsf{P} : \{0,1\}^2 \to \{0,1\}$, which are listed in Table 1 with names for easy reference.

TABLE 1

All possible predicates $\mathsf{P} : \{0,1\}^2 \to \{0,1\}$*, where blank cells denote value* 0*. Predicates* $0x, x0, x1, 1x$ *are determined by a single variable. Predicates* $01, \mathsf{Dicut}, \overline{10}, \overline{01}$ *are satisfied by a single assignment or all but a single one.*

x_1	x_2	$\vec{0}$	nOr	01	$0x$	Dicut	$x0$	Cut	nAnd	And	unCut	$x1$	$\overline{10}$	$1x$	$\overline{01}$	Or	$\vec{1}$
0	0		1		1		1		1		1		1		1		1
0	1			1	1			1	1			1	1			1	1
1	0					1	1	1	1					1	1	1	1
1	1									1	1	1	1	1	1	1	1

We first focus on the case where all the constraints in Π use the same predicate P[1], in which case we can represent the VCSP $\mathcal{I}$ by an edge-weighted digraph $G^{\mathcal{I}} = (V, E, w)$. Each variable in V is represented by a vertex, and each constraint over the pair (v, u) will be represented by a directed edge from v to u, with the same weight as the constraint (formally, $E = \{(v, u) \mid (\langle v, u\rangle, \mathsf{P}) \in \Pi\}$, and abusing notation set edge weights $w(v, u) = w(\langle (v, u), P\rangle))$. This transformation preserves all the information about the VCSP and allows us to make reductions between VCSPs with different predicates P as their sole predicate.

Given a digraph G, a predicate P and a subset $S \subseteq V$, define

$$\mathsf{P}_G(S) := \sum_{(v,u)\in E} \mathsf{P}(\mathbf{1}_S(v), \mathbf{1}_S(u)) \cdot w((v, u)),$$

where $\mathbf{1}_S$ denotes the indicator function. For example, applying this definition to the cut predicate $\mathsf{Cut} : (x, y) \to \mathbf{1}_{\{x\neq y\}}$, we have

$$\mathsf{Cut}_G(S) = \sum_{(v,u)\in E} \mathsf{Cut}(\mathbf{1}_S(v), \mathbf{1}_S(u)) \cdot w((v, u)) = \sum_{(v,u)\in E} |\mathbf{1}_S(v) - \mathbf{1}_S(u)| \cdot w((v, u)),$$

which is just the total weight of the edges crossing the cut S. This matches the definition we gave in the introduction, except for the technical subtlety that G is now a directed graph, which makes no difference for symmetric predicates like Cut. We shall assume henceforth that G is directed.

We shall say that a subinstance G_ϵ is an *ϵ-P-sparsifier* of G if

$$\forall S \subseteq V, \qquad \mathsf{P}_{G_\epsilon}(S) \in (1 \pm \epsilon) \cdot \mathsf{P}_G(S).$$

Observe that given an assignment A for the variables V, we can set $S_A := \{u \mid A(u) = 1\}$. It then holds that $\mathrm{Val}_{\mathcal{I}}(A) = \mathsf{P}_{G^{\mathcal{I}}}(S_A)$, where $G^{\mathcal{I}}$ is the appropriate digraph for the VCSP. As there exists a bijection between such VCSPs and digraphs, we conclude as follows.

Observation 2.1. The existence of an ϵ-P-*sparsifier* $G_\epsilon = (V, E_\epsilon, w_\epsilon)$ for $G^{\mathcal{I}}$ implies the existence of an ϵ-sparsifier $\mathcal{I}_\epsilon$ for $\mathcal{I}$ with $|E_\epsilon|$ constraints.

Note that the converse is true as well, i.e., an ϵ-sparsifier for $\mathcal{I}$ implies the existence of an ϵ-P-sparsifier for $G_{\mathcal{I}}$ of size $|\Pi_\epsilon|$. From now on, we focus on finding an ϵ-P-sparsifier for an arbitrary digraph G (for different choices of the predicate P).

[1]The collection of predicates used in a VCSP is sometimes called its *signature*. In this paper we mainly deal with VCSPs whose signature is of size one.

3. A single predicate. In this section we go over all the predicates $\mathsf{P} : \{0,1\}^2 \to \{0,1\}$ and classify them into sparsifiable and nonsparsifiable predicates; see Theorems 3.5, 3.6, and 3.7. For simplicity, we state our sparsification results as existential, but in fact all these sparsifiers can be computed in polynomial time.

Our main technique is a graph transformation, which is well-known but apparently only in very different contexts. On the face of it, it is not clear which predicates other than Cut do admit nontrivial sparsification. For example, the uncut edges in a graph do not satisfy a key property of cuts that was used in [4] for cut-sparsification (namely, a polynomial bound on the number of near-minimum cuts in a graph), and it is not clear a priori which edges must be included in every sparsifier (again in analogy with cuts, where all bridge edges must be retained), These deficiencies suggest that the edge-sampling approach, which is very effective for cuts [4, 15, 8], would fail for other predicates and may further be viewed as evidence for the impossibility of sparsification. Thus, we were surprised to find out that different predicates can all be analyzed using one simple graph transformation, which appears easy in retrospect and provides a unifying explanation.

In our classification, we appeal to two basic predicates, the first of which is Cut, which is already known to be sparsifiable.

THEOREM 3.1 (see [3]). *For every digraph G and parameter $\epsilon \in (0,1)$, there is an ϵ-Cut-*sparsifier *for G with $O\left(|V|/\epsilon^2\right)$ edges.*

Our second basic predicate is the predicate And, which behaves significantly differently. We call a digraph $G = (V,E)$ *strongly asymmetric* if for every $(v,u) \in E$ it holds that $(u,v) \notin E$.

THEOREM 3.2. *For every strongly asymmetric digraph $G = (V,E,w)$ with strictly positive weights and $\epsilon \in (0,1)$, every ϵ-And-sparsifier $G_\epsilon = (V,E_\epsilon,w_\epsilon)$ must satisfy $E_\epsilon = E$.*

Proof. Let $G_\epsilon = (V,E_\epsilon,w_\epsilon)$ be such a sparsifier, i.e., for every $S \subseteq V$ it holds that $\mathsf{And}_{G_\epsilon}(S) \in (1\pm\epsilon)\cdot \mathsf{And}_G(S)$. Then for every $e = (v,u) \in E$ we must have $(v,u) \in E_\epsilon$, as otherwise for the set $S = \{v,u\}$ it will hold that $\mathsf{And}_{G_\epsilon}(\{v,u\}) = 0$ while $\mathsf{And}_G(\{v,u\}) = w(e) > 0$, a contradiction. ∎

Remark 3.3. For every digraph (which is not necessarily strongly asymmetric), the same proof shows that $|E_\epsilon| \geq \frac{1}{2}|E|$.

Remark 3.4. Our definition of an ϵ-P-sparsifier requires G_ϵ to be a subgraph of G, but we can state Theorem 3.2 in a more general way: For every digraph $G_\epsilon = (V,E_\epsilon,w_\epsilon)$ (not necessarily a subgraph) such that every $S \subseteq V$ satisfies $\mathsf{And}_{G_\epsilon}(S) \in (1\pm\epsilon)\cdot \mathsf{And}_G(S)$ necessarily E_ϵ agrees with E up to the directions of the edges.

Next, we show that every other predicate is similar either to Cut or to And in terms of sparsifability. We describe a reduction that will be useful to show both sparsifability and nonsparsifability. (This reduction is based on a well-known transformation of a given graph, called the "bipartite double cover" (see, e.g., [6]), although we are not aware of its use in the same way.) Let γ be a function that maps a digraph $G = (V,E,w)$ where $V = \{v_1,v_2,\dots,v_n\}$ to a digraph $\gamma(G) = (V^\gamma,E^\gamma,w^\gamma)$ where $V^\gamma = \{v_{-n},\dots,v_{-1},v_1,\dots,v_n\}$, $E^\gamma = \{(v_i,v_{-j}) \mid (v_i,v_j) \in E\}$, $w^\gamma((v_i,v_{-j})) = w((v_i,v_j))$. For every subset $S \subseteq V$, we introduce the notation $-S := \{v_{-i} \mid v_i \in S\}$, $\bar{S} := \{v_i \mid v_i \in V\setminus S\}$ and $-\bar{S} := \{v_{-i} \mid v_i \in V\setminus S\}$. Figure 1 illustrates the effect of γ on an arbitrary set S.

THEOREM 3.5. *For every digraph $G = (V,E,w)$ and $\epsilon \in (0,1)$ there is a subdigraph G_ϵ with $O(|V|/\epsilon^2)$ edges such that for every predicate $\mathsf{P} \in$*

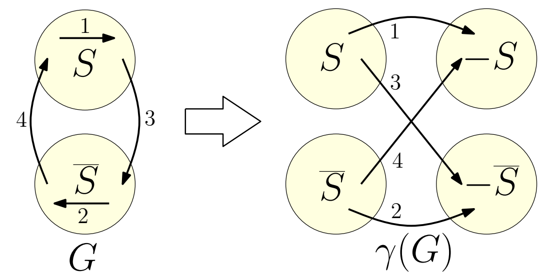

FIG. 1. *The mapping γ applied on G and its effect on an arbitrary $S \subseteq V$. For example, an edge from $v_i \in S$ to $v_j \in \bar{S}$ is represented by an arrow of type 3 and becomes in $\gamma(G)$ an edge from $v_i \in S$ to $v_{-j} \in -\bar{S}$.*

$\{\mathit{Cut}, \mathit{unCut}, \mathit{Or}, \mathit{nAnd}, \overline{10}, \overline{01}, x0, x1, 0x, 1x, \vec{1}, \vec{0}\}$*, the digraph G_ϵ is an ϵ-P-*sparsifier *of G. (Note that G_ϵ does not depend on P.)*

Proof. Given G and ϵ, first construct $\gamma(G)$ as above. Next, apply Theorem 3.1 to obtain for $\gamma(G)$ a cut sparsifier $\gamma(G)_\epsilon = (V^\gamma, E^\gamma_\epsilon \subseteq E^\epsilon, w^\gamma_\epsilon)$, which contains $O(|V^\gamma|/\epsilon^2) = O(|V|/\epsilon^2)$ edges. Now construct a digraph $G_\epsilon = (V, E_\epsilon, w_\epsilon)$ where $E_\epsilon = \{(v_i, v_j) \mid (v_i, v_{-j}) \in E^\gamma_\epsilon\}$ and $w_\epsilon(v_i, v_j) = w^\gamma_\epsilon(v_i, v_{-j})$. Observe that $\gamma(G_\epsilon) = \gamma(G)_\epsilon$, i.e., if we apply γ on G_ϵ we get exactly $\gamma(G)_\epsilon$.

Now suppose that for a predicate P, there is a function $f_P : 2^V \to 2^{V^\gamma}$ such that for every digraph H on the vertex set V, it holds that

$$\forall S \subset V, \qquad \mathsf{P}_H(S) = \mathsf{Cut}_{\gamma(H)}(f_P(S)). \tag{2}$$

Then we could apply (2) twice, first to G_ϵ and then to G, and obtain that

$$\forall S \subset V, \qquad \mathsf{P}_{G_\epsilon}(S) = \mathsf{Cut}_{\gamma(G)_\epsilon}(f_P(S)) \in (1 \pm \epsilon) \cdot \mathsf{Cut}_{\gamma(G)}(f_P(S)) = (1 \pm \epsilon) \cdot \mathsf{P}_G(S).$$

Hence, the existence of such a function f_P implies that G_ϵ is an ϵ-P-sparsifier. And indeed, we can show such f_P for some predicates P, as follows:

- $f_{\mathsf{unCut}}(S) = S \cup -\bar{S}$;
- $f_{\mathsf{Cut}}(S) = S \cup -S$;
- $f_{0x}(S) = \bar{S}$;
- $f_{x0}(S) = -\bar{S}$;
- $f_{x1}(S) = -S$;
- $f_{1x}(S) = S$;
- $f_{\vec{1}}(S) = S \cup \bar{S}$; and
- $f_{\vec{0}}(S) = \emptyset$.

To verify that $f_{\mathsf{unCut}}(S) = S \cup -\bar{S}$ satisfies Equation 2, i.e., that $\mathsf{unCut}_H(S) = \mathsf{Cut}_{\gamma(H)}(S \cup \bar{S})$, observe that both sides consist exactly of the edges of types 1 and 2 in Figure 1. The other predicates can be easily verified similarly, which completes the proof for all $\mathsf{P} \in \{\mathsf{Cut}, \mathsf{unCut}, 0x, x0, x1, 1x, \vec{1}, \vec{0}\}$.

To show that G_ϵ is a sparsifier also for predicates $\mathsf{P} \in \{\mathsf{Or}, \mathsf{nAnd}, \overline{10}, \overline{01}\}$ we need a slightly more general argument. Suppose that for a predicate P, there are functions $f^1_P, f^2_P, f^3_P : 2^V \to 2^{V^\gamma}$ such that for every digraph H on the vertex set V,

$$\mathsf{P}_H(S) = \tfrac{1}{2}\left[\mathsf{Cut}_{\gamma(H)}(f^1_P(S)) + \mathsf{Cut}_{\gamma(H)}(f^2_P(S)) + \mathsf{Cut}_{\gamma(H)}(f^3_P(S))\right]. \tag{3}$$

Then we could apply (3) twice, first to G_ϵ and then to G, and obtain that

$$\begin{aligned}\mathsf{P}_{G_\epsilon}(S) &= \tfrac{1}{2}\left[\mathsf{Cut}_{\gamma(G)_\epsilon}(f_P^1(S)) + \mathsf{Cut}_{\gamma(G)_\epsilon}(f_P^2(S)) + \mathsf{Cut}_{\gamma(G)_\epsilon}(f_P^3(S))\right] \\ &\in (1 \pm \epsilon)\cdot \tfrac{1}{2}\left[\mathsf{Cut}_{\gamma(G)}(f_P^1(S)) + \mathsf{Cut}_{\gamma(G)}(f_P^2(S)) + \mathsf{Cut}_{\gamma(G)}(f_P^3(S))\right] \\ &= (1 \pm \epsilon)\cdot \mathsf{P}_G(S).\end{aligned}$$

Hence, the existence of three such functions will imply that G_ϵ is an ϵ-P-sparsifier. And indeed, we let

- $f^1_{\mathsf{Or}}(S) = S$, $f^2_{\mathsf{Or}}(S) = -S$, $f^3_{\mathsf{Or}}(S) = S \cup -S$;
- $f^1_{\mathsf{nAnd}}(S) = \bar{S}$, $f^2_{\mathsf{nAnd}}(S) = -\bar{S}$, $f^3_{\mathsf{nAnd}}(S) = \bar{S} \cup -\bar{S}$;
- $f^1_{\overline{10}}(S) = \bar{S}$, $f^2_{\overline{10}}(S) = -S$, $f^3_{\overline{10}}(S) = \bar{S} \cup -S$; and
- $f^1_{\overline{01}}(S) = S$, $f^2_{\overline{01}}(S) = -\bar{S}$, $f^3_{\overline{01}}(S) = S \cup -\bar{S}$.

To verify that $f^1_{\mathsf{Or}}, f^2_{\mathsf{Or}}, f^3_{\mathsf{Or}}$ satisfies (3), observe that both sides consist exactly of the edges of types $1, 3, 4$ in Figure 1. The other predicates can be easily verified similarly, which completes the proof for all $\mathsf{P} \in \{\mathsf{Or}, \mathsf{nAnd}, \overline{10}, \overline{01}\}$. ∎

Next, we use γ for a reduction from And to all the remaining predicates. In particular it will imply their "resistance to sparsification."

THEOREM 3.6. *Given parameters n and $m \le \binom{n}{2}$, there is a digraph $G = (V, E, w)$ with $2n$ vertices and m edges such that for every $\epsilon \in (0, 1)$ and every predicate $\mathsf{P} \in \{\mathsf{nOr}, 01, \mathsf{Dicut}, \mathsf{And}\}$, for every ϵ-P-sparsifier $G_\epsilon = (V, E_\epsilon, w_\epsilon)$ of G it holds that that $E_\epsilon = E$. (Note that G does not depend on P.)*

Proof. Let $G = (V, E, w)$ be an arbitrary strongly asymmetric digraph with n vertices, m edges, and strictly positive weights. Let $\gamma(G)$ be the digraph constructed by our reduction. Note that $\gamma(G)$ consist of $2n$ vertices and m edges. $\gamma(G)$ will be the digraph for which we will prove the theorem.

Fix some predicate P. Let $\gamma(G)_\epsilon = (V^\gamma, E^\gamma_\epsilon \subseteq E_\epsilon, w^\gamma_\epsilon)$ be some ϵ-P-sparsifier for $\gamma(G)$. Let $G_\epsilon = (V, E_\epsilon, w_\epsilon)$ be a digraph where $E_\epsilon = \{(v_i, v_j) \mid (v_i, v_{-j}) \in E^\gamma_\epsilon\}$ and $w_\epsilon((v_i, v_j)) = w^\gamma_\epsilon((v_i, v_{-j}))$. Note that $\gamma(G_\epsilon) = \gamma(G)_\epsilon$.

Now suppose that there is a function $f_P : 2^V \to 2^{V^\gamma}$ such that for every digraph H on the vertex set V, it holds that

$$\forall S \subset V, \qquad \mathsf{And}_H(S) = \mathsf{P}_{\gamma(H)}(f_P(S)). \tag{4}$$

Then we could apply (4) twice, first to G_ϵ and then to G, and obtain that

$$\forall S \subset V, \qquad \mathsf{And}_{G_\epsilon}(S) = \mathsf{P}_{\gamma(G)_\epsilon}(f_P(S)) \in (1 \pm \epsilon)\cdot \mathsf{P}_{\gamma(G)}(f_P(S)) = (1 \pm \epsilon)\cdot \mathsf{And}_G(S).$$

Hence, assuming such a function f exists, G_ϵ is an ϵ-And-sparsifier for G. According to Theorem 3.2, necessarily $E_\epsilon = E$, and in particular $E^\gamma_\epsilon = E^\gamma$.

Hence, the existence of such functions f_P for all $\mathsf{P} \in \{\mathsf{nOr}, 01, \mathsf{Dicut}, \mathsf{And}\}$ will imply our theorem. And indeed, we let

- $f_{And}(S) = S \cup -S$;
- $f_{nOr}(S) = \bar{S} \cup -\bar{S}$;
- $f_{Dicut}(S) = S \cup -\bar{S}$; and
- $f_{01}(S) = \bar{S} \cup -S$.

To verify that $f_{Dicut}(S) = S \cup -\bar{S}$ satisfies (4), observe that both sides consist exactly of the edges of type 1 in Figure 1. The other predicates can be easily verified similarly. ∎

We conclude our main theorem, which basically puts together Theorems 3.5 and 3.6.

THEOREM 3.7. *Let* P *be a binary predicate, and let* $\epsilon \in (0,1)$ *be some parameter.*

- *If* P *has a single "1" in its truth table, then there exist a VCSP* $\mathcal{I} = (V, \Pi, w)$ *with a single predicate* P *such that every* ϵ*-*P*-sparsifier of* $\mathcal{I}$ *will have* $\Omega(|V|^2)$ *constraints.*
- *If* P *does not has a single "1" in its truth table, then for every VCSP* $\mathcal{I} = (V, \Pi, w)$ *with single predicate* P*, there exists an* ϵ*-*P*-sparsifier with* $O\left(|V|/\epsilon^2\right)$ *constraints.*

4. Lower bounds (for a single predicate). In this section we will show that Theorem 3.5 is tight. More precisely, we will show that for every $\mathsf{P} \in \{\mathsf{Cut}, \mathsf{unCut}, \mathsf{Or}, \mathsf{nAnd}, \overline{10}, \overline{01}\}$, there exists an n-vertex graph G such that every ϵ-P-sparsifier G_ϵ of G must contain $\Omega(n/\epsilon^2)$ edges.[2] The first step was done by [2], who showed that Theorem 3.1 is tight, i.e., for every n and $\epsilon \in (1/\sqrt{n}, 1)$, there exists an n-vertex graph G such that every ϵ-Cut-sparsifier G_ϵ of G must contain $\Omega(n/\epsilon^2)$ edges. Using our reduction γ in a similar manner to Theorem 3.5, this lower bound can be extended to unCut based on the fact that $\mathsf{Cut}_G(S) = \mathsf{unCut}_{\gamma(G)}\left(S \cup -\bar{S}\right)$. However, γ fails to extend the lower bound to predicates with three 1's in their truth table. To this end, we will define sketching schemes, a variation of sparsification where the goal is to maintain the approximate value of every assignment using a small data structure, possibly without any combinatorial structure; see the definition below. We will use a lower bound on the sketch-size of Cut from [2] to prove the lower bound on the number of edges in a sparsifier (and also on the sketch-size) for Or. The extension to other predicates with three 1's in their truth table is straightforward using γ. Sketching is interesting on its own, and we have further discussion and lower bounds regarding sketching in section 6.3.

Formally, a *sketching scheme* (or a *sketch* in short) is a pair of algorithms $(\mathrm{sk}, \mathrm{est})$. Given a weighted digraph $G = (V, E, w)$ and a predicate P, algorithm sk returns a string sk_G (intuitively, a short encoding of the instance). Given $\mathrm{sk}_\mathcal{I}$ and a subset $S \subseteq V$, algorithm est returns a value (without looking at G) that estimates $\mathsf{P}_G(S)$. We say that it is an ϵ-P-*sketching-scheme* if for every digraph G, and for every subset $S \subseteq V$, $\mathrm{est}(\mathrm{sk}_G, S) \in (1 \pm \epsilon) \cdot \mathsf{P}_G(S)$. The *sketch-size* is $\max_G |\mathrm{sk}_G|$, the maximum length of the encoding string over all the digraphs with n variables, often measured in bits. *sk* might be probabilistic algorithm, but for our purposes it is enough to think only about the deterministic case. Note that an algorithm for constructing ϵ-sparsifiers always provides an ϵ-sketching-scheme, where the sketch-size is asymptotically equal to the number of constraints in the constructed sparsifiers when measured in machine words (and up to logarithmic factors when measured in bits). Sparsification is advantageous over general sketching as it preserves the combinatorial structure of the problem. Nevertheless, one may be interested in constructing sketches as they may potentially require significantly smaller storage.

THEOREM 4.1. *Fix a predicate* $\mathsf{P} \in \{\mathsf{Cut}, \mathsf{unCut}, \mathsf{Or}, \mathsf{nAnd}, \overline{10}\}$*, an integer* n*, and* $\epsilon \in (1/\sqrt{n}, 1)$*. The sketch-size of every* ϵ*-*P*-sketching-scheme on* n *variables is* $\Omega(n/\epsilon^2)$*. Moreover, there is an* n*-vertex digraph* G*, such that every* ϵ*-*P*-sparsifier of* G *has* $\Omega(n/\epsilon^2)$ *edges.*

[2]The other predicates $\{x0, x1, 0x, 1x, \vec{1}, \vec{0}\}$ are kind of trivial in the sense of sparsification. $\vec{0}$ sparsified by the empty graph. $\vec{1}$ can be sparsified using a single edge. $\{x0, x1, 0x, 1x\}$ could be sparsified using n edges.

Proof. We follow the line-of-proof of Theorems 2.3 and 2.4 in [2]. Specifically, they show that the sketch-size of every ϵ-Cut-sketching-scheme is $\Omega(n/\epsilon^2)$ bits, by proving that a certain family $\mathcal{F}$ of n-vertex graphs is hard to sketch and consequently to sparsify. By similar arguments to Theorem 3.5, this lower bound easily extends to unCut. Indeed, recall that $\mathsf{Cut}_G(S) = \mathsf{unCut}_{\gamma(G)}\left(S \cup -\bar{S}\right)$, and thus a ϵ-unCut-sparsifier (or sketch) for $\gamma(G)$ yields an ϵ-Cut-sparsifier (or sketch) for G with the same number of edges (size).

Once we prove the lower bound for predicate Or, a reduction from Or using γ will extend it also to nAnd, $\overline{10}$ and $\overline{01}$, because

$$\mathsf{Or}_G(S) = \mathsf{nAnd}_{\gamma(G)}(\bar{S} \cup -\bar{S}) = \overline{01}_{\gamma(G)}(S \cup -\bar{S}) = \overline{10}_{\gamma(G)}(\bar{S} \cup -S). \tag{5}$$

We will thus focus on the predicate Or. As it is a symmetric predicate, we can work with graphs rather then digraphs. The main observation in our proof is that for every undirected graph $G = (V, E, w)$, if $\deg_G(v)$ denotes the degree of vertex v, then

$$\forall S \subset V, \qquad \mathsf{Cut}_G(S) = 2 \cdot \mathsf{Or}_G(S) - \sum_{v \in S} \deg_G(v). \tag{6}$$

The graph family $\mathcal{F}$ consists of graphs G constructed as follows. Let $s_1, \ldots, s_{n/2} \in \{0,1\}^{1/\epsilon^2}$ be balanced $1/\epsilon^2$ bit-strings (i.e., each s_i has normalized Hamming weight exactly $1/2$), and let the graph G be a disjoint union of the graphs $\{G_j \mid j \in [\epsilon^2 n/2]\}$, where each G_j is a bipartite graph, whose two sides, each of size $1/\epsilon^2$, are denoted $L(G_j)$ and $R(G_j)$. The edges of G are determined by $s_1, \ldots, s_{n/2}$, where each bit string s_i is indicates the adjacency between vertex $i \in \cup_j L(G_j)$ and the vertices in the respective $R(G_j)$. They further observe (in the proof of [2, Theorem 2.4]) that the lower bound holds even if the sketching scheme is relaxed as follows:

1. The estimation is required only for cut queries contained in a single G_j, namely, cut queries $S \cup T$, where $S \subset L(G_j)$ and $T \subset R(G_j)$ for the same j.
2. The estimation achieves additive error μ/ϵ^3, where $\mu = 10^{-4}$ (instead of multiplicative error $1 \pm \epsilon$).

To prove a sketch-size lower bound for a $(\mu\epsilon)$-Or-sketching-scheme $(\mathrm{sk}^{\mathsf{Or}}, \mathrm{est}^{\mathsf{Or}})$, we assume it has sketch-size $s = s(n, \epsilon)$ bits and use it to construct a Cut-sketching-scheme $(\mathrm{sk}^{\mathsf{Cut}}, \mathrm{est}^{\mathsf{Cut}})$ that achieves the estimation properties 1 and 2 on graphs of the aforementioned form and has sketch-size $s + 2n\log(1/\epsilon)$ bits. Then by [2], this sketch-size must be $\Omega(n/\epsilon^2)$, and we conclude that $s = \Omega(n/\epsilon^2)$ as required.

Given a graph $G \in \mathcal{F}$, let $\mathrm{sk}_G^{\mathsf{Cut}}$ be a concatenation of $\mathrm{sk}_G^{\mathsf{Or}}$ and a list of all vertex degrees in G. The degrees in G are bounded by $1/\epsilon^2$, hence the size of $\mathrm{sk}_G^{\mathsf{Cut}}$ is indeed $s + 2n\log(1/\epsilon)$ bits. Given a cut query $S \cup T$ contained in some G_j, define the estimation algorithm (which we now construct for Cut) to be

$$\mathrm{est}^{\mathsf{Cut}}(\mathrm{sk}_G^{\mathsf{Cut}}, S \cup T) := 2 \cdot \mathrm{est}^{\mathsf{Or}}(\mathrm{sk}_G^{\mathsf{Or}}, S \cup T) - \sum_{v \in S \cup T} \deg_G(v). \tag{7}$$

Let us analyze the error of this estimate. First, observe that as in each G_j there are precisely $\frac{1}{2\epsilon^4}$ edges, $\mathsf{Or}_G(S \cup T) \le \frac{1}{2\epsilon^4}$, and thus

$$\mathrm{est}^{\mathsf{Or}}(\mathrm{sk}_G^{\mathsf{Or}}, S \cup T) \in (1 \pm \mu\epsilon) \cdot \mathsf{Or}_G(S \cup T) \subseteq \mathsf{Or}_G(S \cup T) \pm \frac{\mu}{2\epsilon^3} .$$

Plugging this estimate into (7) and then recalling our initial observation (6), we obtain as desired

$$\begin{aligned}\mathrm{est}^{\mathsf{Cut}}(\mathrm{sk}_G^{\mathsf{Cut}}, S\cup T) &\in 2\cdot \mathsf{Or}_G(S\cup T) \pm \frac{\mu}{\epsilon^3} - \sum_{v\in S\cup T}\deg_G(v)\\ &= \mathsf{Cut}_G(S\cup T) \pm \frac{\mu}{\epsilon^3}\ .\end{aligned}$$

To prove a lower bound on the size of an Or-sparsifier, we follow the argument in [2, Theorem 2.4], which shows that given an ϵ-Cut-sparsifier G_ϵ with $s = s(n,\epsilon)$ edges for a graph $G \in \mathcal{F}$, there is a Cut-sparsifier G_μ of G_ϵ, with additive error $\mu/2\epsilon^3$, such that G_μ has only integer weights and henceforth can be encoded using $O(s(\mu^{-2} + \log(\epsilon^{-2}n/s)))$ bits. In fact, there is nothing special here about Cut. The same proof will work (with the same properties) for predicate Or, assuming a sparsifier is required to be a subgraph (to remove this restriction, just erase all the edges between G_j to G_i for $i \neq j$, which adds only a small additive error).

Now suppose that every graph G of the form specified above admits a $\frac{\mu}{2}\epsilon$-Or-sparsifier G_ϵ with s edges. Then as explained above (about repeating the argument of [2]) there is a graph G_μ that sparsifies G_ϵ with additive error $\mu/2\epsilon^3$ and can be encoded by a string $\mathcal{I}_G$ of size $O(s\log(\epsilon^{-2}n/s))$ bits (recall that μ is a constant). Use it to construct a Cut-sketching-scheme with additive error μ/ϵ^3 as follows. Given the graph G, set $\mathrm{sk}_G^{\mathsf{Cut}}$ to be the concatenation of $\mathcal{I}_G$ and a list of the degrees of all the vertices in G. Then $|\mathcal{I}_G| = O(s\log(\epsilon^{-2}n/s)) + 2n\log(1/\epsilon)$. For a cut query $S\cup T$ contained in some G_j, define the estimation algorithm (using the Or sparsifier) to be

$$\mathrm{est}^{\mathsf{Cut}}(\mathrm{sk}_G^{\mathsf{Cut}}, S\cup T) := 2\cdot \mathsf{Or}_{G_\mu}(S\cup T) - \sum_{v\in S\cup T}\deg_G(v).$$

Then we can again analyze it by plugging the above error bounds and then using (6),

$$\begin{aligned}\mathrm{est}^{\mathsf{Cut}}(\mathrm{sk}_G^{\mathsf{Cut}}, S\cup T) &\in 2\cdot \mathsf{Or}_{G_\epsilon}(S\cup T) \pm \frac{\mu}{2\epsilon^3} - \sum_{v\in S\cup T}\deg_G(v)\\ &\in 2\cdot \mathsf{Or}_G(S\cup T) \pm \frac{\mu}{\epsilon^3} - \sum_{v\in S\cup T}\deg_G(v)\\ &= \mathsf{Cut}_G(S\cup T) \pm \frac{\mu}{\epsilon^3}\ .\end{aligned}$$

By [2], the sketch-size must be $|\mathcal{I}_G| = \Omega(n/\epsilon^2)$, hence $s = \Omega(n/\epsilon^2)$ (for at least one graph $G \in \mathcal{F}$) as required. ∎

5. Multiple predicates and applications. In this section we extend Theorem 3.5 to VCSPs using multiple types of predicates. In particular, we prove sparsifability for some classical problems. Again, our sparsification results are stated as existential bounds, but these sparsifiers can actually be computed in polynomial time.

THEOREM 5.1. *For every $\epsilon \in (0,1)$ and a VCSP (V,Π,w) whose constraints $\langle (v,u), P\rangle \in \Pi$ all satisfy $P \notin \{\mathit{nOr}, 01, \mathit{Dicut}, \mathit{And}\}$, there exists an ϵ-sparsifier for $\mathcal{I}$ with $O(|V|/\epsilon^2)$ constraints.*

This bound is tight, according to Theorem 4.1. We prove it by a straightforward application of Theorem 3.5. Partition $\mathcal{I}$ to disjoint VCSPs according to the predicates in the constraints, and then for each sub-VCSP find an ϵ-sparsifier using Theorem 3.5. The union of this sparsifiers is an ϵ-sparsifier for $\mathcal{I}$. A formal proof follows.

Proof of Theorem 5.1. For each predicate P, let $\Pi^P = \{\pi \in \Pi \mid \pi = \langle (v,u), \mathsf{P}\rangle\}$. Note that $\{\Pi^P\}$ forms a partition of Π. For each P, let $\mathcal{I}^P = (V, \Pi^P, w^P)$, where w^P is the restriction of w to Π^P. Let $\mathcal{I}^P_\epsilon = (V, \Pi^P_\epsilon, w^P_\epsilon)$ be an ϵ-P-sparsifier for $\mathcal{I}^P$ with $|\Pi^P_\epsilon| = O(|V|/\epsilon^2)$ constraints according to Theorem 3.5 (recall that $\mathsf{P} \notin \{\mathsf{nOr}, 01, \mathsf{Dicut}, \mathsf{And}\}$). Set $\mathcal{I}_\epsilon = (V, \Pi_\epsilon, w_\epsilon)$, $\Pi_\epsilon = \bigcup_P \Pi^P_\epsilon$ and $w_\epsilon = \bigcup_P w^P_\epsilon$. For every assignment A,

$$\begin{aligned}
\mathrm{Val}_{\mathcal{I}_\epsilon}(A) &= \sum_{\pi_i \in \Pi_\epsilon} w_\epsilon(\pi_i) \cdot p_i(A(v_i), A(u_i)) \\
&= \sum_{\mathsf{P}} \sum_{\pi_i \in \Pi^P_\epsilon} w^P_\epsilon(\pi_i) \cdot \mathsf{P}(A(v_i), A(u_i)) \\
&\in (1 \pm \epsilon) \cdot \sum_{P} \sum_{\pi_i \in \Pi^P} w^P(\pi_i) \cdot \mathsf{P}(A(v_i), A(u_i)) \\
&= (1 \pm \epsilon) \cdot \sum_{\pi_i \in \Pi} w(\pi_i) \cdot p_i(A(v_i), A(u_i)) \\
&= (1 \pm \epsilon) \cdot \mathrm{Val}_{\mathcal{I}}(A),
\end{aligned}$$

and note that indeed $|\Pi_\epsilon| \le O\left(n/\epsilon^2\right)$. ∎

$\mathsf{2SAT}$ (boolean satisfiability problem over constraints with two variables) can be viewed as a VCSP which uses only the predicates Or, nAnd, $\overline{10}$, and $\overline{01}$. By Theorem 5.1, for every $\mathsf{2SAT}$ formula Φ over n variables, and for every $\epsilon \in (0,1)$, there is a sub-formula Φ_ϵ with $O(n/\epsilon^2)$ clauses, such that Φ and Φ_ϵ have the same value for every assignment up to factor $1+\epsilon$.[3]

$\mathsf{2LIN}$ is a system of linear equations (modulo 2), where each equation contains two variables and has a nonnegative weight. Notice that the equation $x+y=1$ is a constraint using the Cut predicate, while the equation $x+y=0$ is a constraint using the unCut predicate. By Theorem 5.1, if n denotes the number of variables, then for every $\epsilon \in (0,1)$ we can construct a sparsifier with only $O(n/\epsilon^2)$ equations (i.e., a reweighted subset of equations, such that on every assignment it agrees with the original system up to factor $1+\epsilon$).

We note that by our lower bound (Theorem 4.1), there are instances of $\mathsf{2SAT}$ ($\mathsf{2LIN}$) for which every ϵ-sparsifier must contain $\Omega(n/\epsilon^2)$ clauses (equations).

6. Further directions. Based on the past experience of cut sparsification in graphs—which has been extremely successful in terms of techniques, applications, extensions, and mathematical connections—we expect VCSP sparsification to have many benefits. A challenging direction is to identify which predicates admit sparsification, and our results make the first strides in this direction.

We now discuss potential extensions to our results in the previous sections (which characterize two-variable predicates over a boolean alphabet). We first consider predicates with more variables, and in particular show sparsification for k-SAT formulas, in section 6.1. We then consider predicates with large alphabets in section 6.2, showing in particular a sparsifier construction for k-Cut and that linear equations (modulo $k \ge 3$) are not sparsifiable. We also consider sketching schemes; notably we discuss a looser sketching model called *for-each* in section 6.3. Finally, we study *spectral* sparsification for unCut, a notion that preserves some algebraic properties in addition to the "uncuts" in section 6.4.

[3]We use here the version of $\mathsf{2SAT}$ where each clause has weight and every assignment has value, rather than the version when we only ask whether there is an assignment that satisfies all the clauses.

6.1. Predicates over more variables and k-SAT. It is natural to ask for the best bounds on the size of ϵ-P-sparsifiers for different predicates $\mathsf{P} : \{0,1\}^k \to \{0,1\}$. A first step toward answering this question was already done by [11].

THEOREM 6.1 (see [11]). *For every hypergraph $H = (V,E,w)$ with hyperedges containing at most r vertices, and $\epsilon \in (0,1)$, there is a reweighted subhypergraph H_ϵ with $O(n(r + \log n)/\epsilon^2)$ hyperedges such that*

$$\forall S \subseteq V, \quad \mathsf{Cut}_{H_\epsilon}(S) \in (1 \pm \epsilon) \cdot \mathsf{Cut}_H(S).$$

Here we say that a hyperedge e is *cut* by S if $S \cap e \notin \{\emptyset, e\}$ (i.e., not all the vertices in e are in the same side). Observe that Cut is equivalent to the predicate NAE (not all equal). In particular Theorem 6.1 implies that for every VCSP using only NAE, there is an ϵ-sparsifier with $O(n(r + \log n)/\epsilon^2)$ constraints.

A k-SAT is essentially a VCSP that uses only predicates with a single 0 in their truth table. Kogan and Krauthgamer [11] use Theorem 6.1 to construct an ϵ-sketching-scheme with sketch-size $\tilde{O}(nk/\epsilon^2)$ for k-SAT formulas (i.e., only for VCSPs of this particular form). We observe that their sketching scheme can be further used to construct an ϵ-sparsfier, as follows.

First, recall how the sketching scheme of [11] works. Given a k-SAT formula $\Phi = (V, \mathcal{C}, w)$ (variables, clauses, weight over $\mathcal{C}$), construct a hypergraph H on vertex set $V \cup -V \cup \{f\}$. We associate the literal v_i with vertex v_i, associate the literal $\neg v_i$ with vertex v_{-i}, and use f to represent the "false." Each clause becomes a hyperedge consisting of f and (the vertices associated with) the literals in $\mathcal{C}$ (for example, $v_5 \vee \neg v_7 \vee v_{12}$ becomes $\{f, v_5, v_{-7}, v_{12}\}$). Observe that given a truth assignment $A : V \to \{0,1\}$, if we define $S_A := \{u \mid A(u) = 0\}$, then $\mathrm{Val}_\Phi(A) = \mathsf{Cut}_H(S_A \cup \{f\})$, and using Theorem 6.1 this provides a sketching scheme. Moreover, given an ϵ-Cut-sparsifier H_ϵ for H, let Φ_ϵ be the formula which has only the clauses associated with edges that "survived" the sparsification, with the same weight. Notice that for every assignment A,

$$\mathrm{Val}_{\Phi_\epsilon}(A) = \mathsf{Cut}_{H_\epsilon}(S_A \cup \{f\}) \in (1 \pm \epsilon) \cdot \mathsf{Cut}_H(S_A \cup \{f\}) = (1 \pm \epsilon) \cdot \mathrm{Val}_\Phi(A).$$

THEOREM 6.2. *Given k-SAT formula Φ over n variables and parameter $\epsilon \in (0,1)$, there is an ϵ-sparsifier subformula ϕ_ϵ with $O(n(k + \log n)/\epsilon^2)$ clauses.*

In contrast, we are not aware of any nontrivial sparsification result for the parity predicate (on $k \geq 3$ boolean variables), and this remains an interesting open problem.

6.2. Predicates over larger alphabets. Our results deal only with predicates that get two input values in $\{0,1\}$. A natural generalization is to sparsify a VCSP that uses a predicate over an alphabet of size k, i.e., $\mathsf{P} : [k] \times [k] \to \{0,1\}$, where $[k] := \{0, 1, \ldots, k-1\}$. One predicate that we can easily sparsify is NE (not-equal), which is satisfied if the two constrained variables are assigned different values. Indeed, in the graphs language, this is called a k-Cut, where the value of a partition $(S_0, \ldots, S_{k-1})$ of the vertices is the total weight of all edges with endpoints in different parts. It turns out that the ϵ-Cut-sparsifier is in particular an ϵ-k-Cut-sparsifier, using the following well-known double-counting argument:

$$\begin{aligned}\mathsf{k\text{-}Cut}_{G_\epsilon}(S_0,\ldots,S_{k-1}) &= \frac{1}{2}\cdot\left[\mathsf{Cut}_{G_\epsilon}\left(S_0,\overline{S_0}\right)+\cdots+\mathsf{Cut}_{G_\epsilon}\left(S_{k-1},\overline{S_{k-1}}\right)\right]\\ &\in (1\pm\epsilon)\cdot\frac{1}{2}\cdot\left[\mathsf{Cut}_{G}\left(S_0,\overline{S_0}\right)+\cdots+\mathsf{Cut}_{G}\left(S_{k-1},\overline{S_{k-1}}\right)\right]\\ &= (1\pm\epsilon)\cdot\mathsf{k\text{-}Cut}_G(S_0,\ldots,S_{k-1}).\end{aligned}$$

In contrast, linear equation predicates are nonsparsifiable for alphabet $[k]$ of size $k\geq 3$. Specifically, for $a\in[k]$, let the predicate Sum_a be satisfied by $x,y\in[k]$ iff $x+y=a\pmod k$. Then for every positively weighted digraph $G=(V,E,w)$, and every $\epsilon\in(0,1)$, $a\in[k]$, every Sum_a-ϵ-sparsifier $G_\epsilon=(V,E_\epsilon,w_\epsilon)$ of G must have $E=E_\epsilon$. The argument is similar to the proof of Theorem 3.2. Assume for contradiction there exist $e\in E\setminus E_\epsilon$. Choose $x,y,z\in[k]$ that satisfy $x+y=a$, however the three sums $z+x$, $z+y$, $z+z$ are all not equal to a (modulo k); this is clearly possible for $k\geq 4$ and easily verified by case analysis for $k=3$. Consider an assignment where the endpoints of e have values x and y, respectively, and all other vertices have value z. Under this assignment, the value of G is $w(e)>0$, while the value of G_ϵ is zero, a contradiction.

6.3. Sketching. In Theorem 4.1 we showed that for every predicate $\mathsf{P}\in\{\mathsf{Cut},\mathsf{unCut},\mathsf{Or},\mathsf{nAnd},\overline{10}\}$, the sketch-size of every ϵ-P-sketching-scheme is $\Omega(n/\epsilon^2)$.

Let us now address predicates with a single 1 in their truth table. In the spirit of the proof of Theorem 3.2, given encoding sk_G by an ϵ-And-sketching-scheme we can completely restore the graph G. As there are $2^{\binom{n}{2}}$ different graphs, the sketch-size of every ϵ-And-sketching-scheme is at least $\Omega(n^2)$ bits. Imitating the proof of Theorem 3.6, we can extend this lower bound to Dicut, 01, and 10.

For-each sketches. In order to reduce storage space of a sketch, one might weaken the requirements even further and allow the sketch to give a good approximation only with high probability. A *for-each sketching scheme* is a pair of algorithms $(\mathrm{sk},\mathrm{est})$; algorithm sk is a randomized algorithm that given a graph G returns a string sk_G, whose distribution we denote by $\mathcal{D}_G$; algorithm est is given such a string sk_G and a subset $S\subseteq V$ and returns (deterministically) a value $\mathrm{est}(\mathrm{sk}_G,S)$. We say that it is an (ϵ,δ)-*P-sketching-scheme* if

$$\forall G=(V,E,w),\forall S\subseteq V,\quad \Pr_{\mathrm{sk}_G\in\mathcal{D}_G}\left[\mathrm{est}(\mathrm{sk}_G,S)\in(1\pm\epsilon)\cdot\mathsf{P}_G(S)\right]\geq 1-\delta\ .$$

In [2], it was showed that if we consider n-vertex graphs with weights only in the range $[1,W]$, then there is an $(\epsilon,1/\mathrm{poly}(n))$-Cut-sketching-scheme with sketch-size $\tilde{O}\left(n\epsilon^{-1}\cdot\log\log W\right)$ bits. Imitating Theorem 3.5, we can construct $(\epsilon,1/\mathrm{poly}(n))$-P-sketching-scheme with the same sketch-size for every predicate P whose truth table does not have a single 1 (and weights restricted to the range $[1,W]$). A nearly matching lower bound by [2] shows that for every $\epsilon\in(2/n,1/2)$, every $(\epsilon,1/10)$-Cut-sketching-scheme must have sketch-size $\Omega(n/\epsilon)$. Using γ, this lower bound can be extended to unCut. This technique does not work for predicates with three 1's in their truth table. Fortunately, we can duplicate the proof of [2] while replacing Cut by Or and using the fact that for every two vertices v,u in the graph G, it holds that $\mathsf{Or}(\{v\})+\mathsf{Or}(\{u\})-\mathsf{Or}(\{v,u\})=\mathbf{1}_{\{\{u,v\}\in E\}}$. We omit the details of this straightforward argument. A reduction from Or using γ and (5) will extend the lower bound also to nAnd,$\overline{10}$ and $\overline{01}$.

Given a sketch sk_G (i.e., one sample from distribution $\mathcal{D}_G$) which encodes an (ϵ, δ)-And-sketching-scheme, one can reconstruct every edge of G (every bit of the adjacency matrix) with constant probability. Standard information-theoretical arguments (indexing problem) imply that the sketch-size of every (ϵ, δ)-And-sketching-scheme is $\Omega(n^2)$ bits. Using γ we can extend this lower bound to Dicut, 01 and 10.

6.4. unCut spectral sparsifiers. Given an undirected n-vertex graph $G = (V, E, w)$, the Laplacian matrix is defined as $L_G = D_G - A_G$, where A_G is the adjacency matrix (i.e., $A_{i,j} = w_{i,j} = w(\{v_i, v_j\})$) and D_G is a diagonal matrix of degrees (i.e., $D_{i,i} = \sum_{j \neq i} w_{i,j}$ and for $i \neq j$, $D_{i,j} = 0$). For every $x \in \mathbb{R}^n$ it holds that $x^t L_G x = \sum_{\{v_i,v_j\} \in E} w_{i,j} \cdot (x_i - x_j)^2$. In particular, for $\mathbf{1}_S$ the indicator vector of some subset $S \subseteq V$ it holds that $\mathbf{1}_S^t L_G \mathbf{1}_S = \mathsf{Cut}_G(S)$. A subgraph H of G is called an *ϵ-spectral-sparsifier* of G if

$$\forall x \in \mathbb{R}^n, \quad x^t L_H x \in (1 \pm \epsilon) \cdot x^t L_G x \ .$$

Note that an ϵ-spectral-sparsifier is in particular an ϵ-Cut-sparsifier. Nonetheless, spectral sparsifiers preserve additional properties such as the eigenvalues of the Laplacian matrix (approximately). Batson, Spielman, and Srivastava [3] showed that every graph admits an ϵ-spectral-sparsifier with $O(n/\epsilon^2)$ edges.

DEFINITION 6.3. *Given a graph G, we call $U_G = (D_G + A_G)$ the* negated Laplacian *of G. Given a subset $S \subseteq V$, let $\phi_S \in \mathbb{R}^n$ be a vector such that $\phi_{S,i} = 1$ if $v_i \in S$ and $\phi_{S,i} = -1$ otherwise.*

One can verify that for arbitrary $x \in \mathbb{R}^n$,

$$x^t U_G x = \sum_{i<j} w_{i,j} \cdot (x_i + x_j)^2 \ .$$

In particular, for every subset $S \subseteq V$, it holds that

$$\phi_S^t U_G \phi_S = 4 \cdot \mathsf{unCut}_G(S) \ .$$

Next, we will show how we can use U_G to construct an unCut-sparsifier G_ϵ (in an alternative way to Theorem 3.5) such that U_{G_ϵ} has (approximately) the same eigenvalues as U_G. A matrix $M \in \mathbb{R}^{n \times n}$ is called *balanced symmetric diagonally dominant* (*BSDD*) if $M = M^t$ and for every index i, $M_{i,i} = \sum_{j \neq i} |M_{i,j}|$. Note that L_G and U_G are both BSDD. A matrix M' is *governed* by M if whenever $M'_{i,j} \neq 0$, also $M_{i,j} \neq 0$ and has the same sign. Note that if H is a subgraph of G, then U_H is governed by U_G. A matrix M' is called an *ϵ-spectral-sparsifier* of M if M' is governed by M and

$$\forall x \in \mathbb{R}^n, \quad x^t M' x \in (1 \pm \epsilon) \cdot x^t M x \ .$$

The following was implicitly shown in [2].

THEOREM 6.4 (see [2]). *Given BSDD matrix $M \in \mathbb{R}^{n \times n}$ and parameter $\epsilon \in (0, 1)$, there is an ϵ-spectral-sparsifier M' for M, where M' is BSDD matrix with $O(n/\epsilon^2)$ nonzero entries.*

Fix a graph G and parameter ϵ; according to Theorem 6.4, there is a BSDD balanced matrix H with $O(n/\epsilon^2)$ nonzero entries, which is a ϵ-spectral-sparsifier for U_G. Moreover, H is governed by U_G. These properties define a graph G_ϵ such that $U_{G_\epsilon} = H$. In particular G_ϵ is an ϵ-unCut-sparsifier of G with $O(n/\epsilon^2)$ edges.

REFERENCES

[1] K. J. Ahn and S. Guha, *Graph sparsification in the semi-streaming model*, in 36th International Colloquium on Automata, Languages and Programming, Lecture Notes in Comput. Sci. 5556, Springer-Verlag, Berlin, 2009, pp. 328–338, https://doi.org/10.1007/978-3-642-02930-1_27.

[2] A. Andoni, J. Chen, R. Krauthgamer, B. Qin, D. P. Woodruff, and Q. Zhang, *On sketching quadratic forms*, in Proceedings of ITCS'16, ACM, 2016, pp. 311–319, https://doi.org/10.1145/2840728.2840753.

[3] J. D. Batson, D. A. Spielman, and N. Srivastava, *Twice-Ramanujan sparsifiers*, SIAM Rev., 56 (2014), pp. 315–334, https://doi.org/10.1137/130949117.

[4] A. A. Benczúr and D. R. Karger, *Approximating* s-t *minimum cuts in* $\tilde{O}(n^2)$ *time*, in Proceedings of the 28th Annual ACM Symposium on Theory of Computing, ACM, 1996, pp. 47–55, https://doi.org/10.1145/237814.237827.

[5] A. A. Benczúr and D. R. Karger, *Randomized Approximation Schemes for Cuts and Flows in Capacitated Graphs*, CoRR cs.DS/0207078, https://arXiv.org/abs/cs/0207078, 2002.

[6] R. A. Brualdi, F. Harary, and Z. Miller, *Bigraphs versus digraphs via matrices*, J. Graph Theory, 4 (1980), pp. 51–73, https://doi.org/10.1002/jgt.3190040107.

[7] M. K. de Carli Silva, N. J. A. Harvey, and C. M. Sato, *Sparse Sums of Positive Semidefinite Matrices*, CoRR abs/1107.0088, https://arXiv.org/abs/1107.0088, 2011.

[8] W. S. Fung, R. Hariharan, N. J. Harvey, and D. Panigrahi, *A general framework for graph sparsification*, in Proceedings of the 43rd Annual ACM Symposium on Theory of Computing, ACM, 2011, pp. 71–80, https://doi.org/10.1145/1993636.1993647.

[9] M. Kapralov and R. Panigrahy, *Spectral sparsification via random spanners*, in Proceedings of the 3rd Innovations in Theoretical Computer Science Conference, ACM, 2012, pp. 393–398, https://doi.org/10.1145/2090236.2090267.

[10] D. R. Karger and M. S. Levine, *Random sampling in residual graphs*, in Proceedings of the Symposium on Theory of Computing, 2002, pp. 63–66.

[11] D. Kogan and R. Krauthgamer, *Sketching cuts in graphs and hypergraphs*, in Proceedings of the Conference on Innovations in Theoretical Computer Science, ACM, 2015, pp. 367–376, https://doi.org/10.1145/2688073.2688093.

[12] A. Madry, *Fast approximation algorithms for cut-based problems in undirected graphs*, in Proceedings of the Symposium on Foundations of Computer Science, IEEE, 2010, pp. 245–254.

[13] I. Newman and Y. Rabinovich, *On multiplicative* λ*-approximations and some geometric applications*, SIAM J. Comput., 42 (2013), pp. 855–883, https://doi.org/10.1137/100801809.

[14] J. Sherman, *Breaking the multicommodity flow barrier for* $O(\sqrt{\log n})$*-approximations to sparsest cut*, in Proceedings of the Symposium on Foundations of Computer Science, 2009, pp. 363–372.

[15] D. A. Spielman and N. Srivastava, *Graph sparsification by effective resistances*, SIAM J. Comput., 40 (2011), pp. 1913–1926, https://doi.org/10.1137/080734029.

[16] D. A. Spielman and S.-H. Teng, *Nearly-linear time algorithms for graph partitioning, graph sparsification, and solving linear systems*, in Proceedings of the 36th Annual ACM Symposium on Theory of Computing, ACM, 2004, pp. 81–90, https://doi.org/10.1145/1007352.1007372.

[17] D. A. Spielman and S.-H. Teng, *Spectral sparsification of graphs*, SIAM J. Comput., 40 (2011), pp. 981–1025, https://doi.org/10.1137/08074489X.

ניתנים לדילול. אנחנו מראים שאם מס' ההשמות עליהן פרדיקט מקבל ערך אמת אינה 1, אז ניתן לדלל אותו ולייצג אותו בעזרת $O(n/\epsilon^2)$ פרדיקטים ממושקלים מחדש. במקרה שיש בדיוק השמה אחת שמקבל ערך אמת, אז לא ניתן לדלל וצריך להשתמש בכל ה $O(n^2)$ פרדיקרטים האפשריים.

היא טובה ובפרט יש לנו עיוות ממוצע קבוע. אנחנו מראים ש2 המושגים האלה שקולים. כלומר קיום של שיכוני תעדוף גורר שיכוני קנה מידה, וההפך. בהינתן גרף ממושקל, אנחנו משתמשים בשקילות זו ובונים עץ פורש של הגרף אשר משקלו לכל היותר $\rho + 1$ פעמים משקל העץ הפורש הקל ביותר, וכן העיוות הממוצע בו הינו $O(\frac{1}{\rho})$, וזאת לכל פרמטר $\rho \in (0,1)$ שנבחר מראש.

לאחר מכן אנו פונים לחקור את בעיית מחיקת נקודות שטיינר. בהינתן גרף ממושקל $G = (V, E, w)$ ותת קבוצה של טרמינלים $K \subseteq V$ בגודל k. המטרה היא למצוא מינור M של G עם הטרמינלים כקודקודיו, אשר משמר את כל המרחקים בין הטרמינלים, עד כדי עיוות כלשהו. השאלה שנשאלת כאן ברקע, היא האם על ידי הוספת קודקודי שטיינר ניתן להעשיר משמעותית את הגיאומטריה של משפחה מסויימת. בתור דוגמא ניתן לבחון בין 2 משפחות של גיאומטריות. משפחה ראשונה הינה כל הגיאומטריות הנוצרות על ידי k קודקודים בגרף מישורים בגודל k. משפחה שניה הינה כל הגיאומטריות המתקבלות מגרפים מישוריים בעלי מספר גדול ככל שנרצה של קודקודים, כאשר אנחנו מסתכלים רק על המרחקים בין k טרמינלים. האם המשפחות הללו שונות או שניתן לשכן כל הגרף מהמשפחה השניה לגרף במשפחה הראשונה עם עיוות קבוע? התרומה שלנו לבעיית מחיקת נקודות שטיינר הינה חסם עליות חדש של $O(\log k)$ אשר משפר את החסם הקודם של $O(\log^2 k)$. החסם שלנו תופס לגרפים כלליים, אך הוא גם החסם הטוב ביותר היודע לגרפים מישוריים. בפרט $O(\log k)$ הינו החסם הטוב ביותר הידוע לשאלה על העושר הגיאומטרי של משפחות.

הנושא האחרון שמופיע בתזה מדבר על דילול. בעבודה מפורסמת על חתכים, מראים שבהינתן גרף על n קודקודים, ניתן לבנות לו מדלל בגודל $O(n/\epsilon^2)$ אשר משמר את ערכי החתכים עד כדי $1 \pm \epsilon$. אנחנו מכלילים תוצאה זו לפרדיקטים נוספים מעבר לחתכים. אנחנו מגדירים פרדיקט בתור פונקציה מ2 משתנים ל $\{0,1\}$ מופע של תוכנית אילוצים הינו אוסף פרדיקטים ממושקל. בהינתן השמה, המשקל של ההשמה הינה סכום המשקלים של הפרדיקטים המסופקים. בהינתן תוכנית אילוצים, המטרה היא לדלל את התוכנית כך שמס' הפרדיקטים יהיה קטן, אך המשקל של כל ההשמות ישמר עד כדי $1 \pm \epsilon$. התוצאה המרכזית כאן הינה קטלוג של איזה פרדיקטנים

תקציר

שיכון זוהי פונקציה בין שני מרחבים מטריים, אשר משמרת מרחקים בקירוב. לעיתים קרובות המרחב המארח אליו משכנים הינו פשוט, או בעל תכונות מועילות אחרות. שיכונים זוהי שיטה אלגוריתמית אשר זכתה להצלחה מרובה. בפרט נמצאו לה שימושים לאלגוריתמי קירוב, אלגוריתמי אונליין, אלגוריתמים מבוזרים ועוד. בשיכונים הקלאסיים יש מגבלה משמעותית. העיוות, דהיינו עד כמה אנחנו מצליחים לשמר מרחקים, בדרך כלל תלוי במספר הנקודות במרחב אותו אנו מעוניינים לשכן. לדוגמא אם יש לנו n נקודות, במקרים רבים העיוות יהיה $O(\log n)$. במילים אחרות, איכות השיכון נמדדת לפי הביצועיים של זוג הנקודות הגרוע ביותר, בעוד שיתכן שהעיוות הטיפוסי טוב בהרבה.

אנחנו חוקרים מדדים מעודנים יותר של מושג העיוות. אנחנו מגדירים שיכוני תיעדוף. כאן, בנוסף למרחב המטרי אנחנו מקבלים סדר על הנקודות $X = \{x_1, \dots, x_n\}$, המטרה היא למצוא שיכון כך שמרחקים הכוללים את הנקודה x_j יהיה תלוי בj ולא בגודל המרחב הכולל n . אף על פי שאנו דורשים הבטחה חזקה יותר על העיוות, ברוב המקרים, גם אם נמדוד את השיכונים שלנו ביחס למדד של הזוג הגרוע ביותר, הביצועים של השיכונים שלנו טובים לא פחות מהשיכונים הקלאסיים. כדגומא למספר תוצאות שהוכחנו, הראנו שכל מרחב מטרי עם n נקודות משתכן למרחב אוקלידי עם עיוות תעדוף $O(\log j)$. כלומר, העיוות על הזוג $\{x_i, x_j\}$ יהיה חסום על ידי $O(\log(\max\{i,j\}))$. דוגמא נוספת היא שיכון של מרחב מטרי לתוך התפלגות של עצים דומיננטיים עם תוחלת עיוות $O(\log j)$.

מדד מעודן נוסף בו אנו מתעניינים נקרא שיכון קנה מידה. כאן אנו דורשים שלכל פרמטר $\epsilon \in (0,1)$, העיוות של כל הזוגות, פרט אולי לחלק יחסי בגודל ϵ יהנו מהבטחת עיוות טובה (כפונקציה של הפרמטר ϵ). שיכון קנה מידה גורר בפרט עיוות ממוצע קבוע. ממבט ראשוני, שיכוני תעדוף ושיכוני קנה מידה נראה שונים ביותר. בשיכוני תעדוף יש לנו אפשרות לבחור קבוצה של נקודות שיהנו מעיוות קטן מאד, אך ההתנהגות הטיפוסית יכולה להיות בסך הכל די גרועה. לעומת זאת, בשיכוני קנה מידה אין לנו שום שליטה לגבי איזה זוגות יענו מהבטחה טובה על העיוות, אך ההתנהגות הטיפוסית

אני, ארנולד פילצר החתום למטה, מצהיר בזאת:

- חיברתי את חיבורי בעצמי, להוציא עזרת ההדרכה שקיבלתי מאת המנחים.
- החומר המדעי הנכלל בעבודה זו הינו פרי מחקרי מתקופת היותי תלמיד מחקר.
- בעבודת מחקר זאת נכלל חומר מקורי שהוא פרי שיתוף פעולה עם אחרים. בפירוט:
  התוצאות בפרק 1 נעשו בשיתוף עם מיכאל אלקין ועופר ניימן.
  התוצאות בפרק 3 נעשו בשיתוף עם רוברט קראוטגמר.
  תוצאות בפרק 4 נעשו בשיתוף עם יאיר ברטל ועופר ניימן.

תאריך	שם	חתימה
26-Mar-2019	**ארנולד פילצר**	

העבודה נעשתה בהדרכת

פרופסור עופר ניימן ו פרופסור רוברט קראוטגמר

במחלקה למדעי המחשב

בפקולטה למדעי הטבע

אוניברסיטת בן גוריון בנגב

מחקר לשם מילוי חלקי של הדרישות לקבלת תואר "דוקטור לפילוסופיה"

על עידונים של מושג העיוות בשיכוני מטריקות

מאת

ארנולד פילצר

הוגש לסינאט אוניברסיטת בן גוריון בנגב

אושר על ידי:

פרופסור עופר ניימן
מנחה

פרופסור רוברט קראוטגמר
מנחה

אדר ב' תשע"ט מרץ 2019

באר שבע

אוניברסיטת בן גוריון בנגב

מחקר לשם מילוי חלקי של הדרישות לקבלת תואר "דוקטור לפילוסופיה"

על עידונים של מושג העיוות בשיכוני מטריקות

מאת

ארנולד פילצר

הוגש לסינאט אוניברסיטת בן גוריון בנגב

אדר ב' תשע"ט — מרץ 2019

באר שבע