Spectral Sparsification of Metrics and Kernels

Kent Quanrud*

Abstract

A set of n points in a geometric space implicitly induces a complete graph where the weight of an edge between two points is a function of the distance between the endpoints. There are many natural problems that arise from such a geometric graph and many standard geometric problems can be recast as simple properties of this graph. A basic algorithmic obstacle that arises is that the explicit size of the graph is quadratic in the size of the input. There is a long line of research overcoming this obstacle in low-dimensional spaces, as well as some positive results in high-dimensional and more abstract models for specific applications. Here we consider graph problems in general and address the issue of constructing the geometric graph. Rather than constructing these graphs exactly, we ask if it is possible to explicitly construct a sparse approximation of these geometric graphs in nearly linear time.

We consider geometric graphs where the edge weights are given as either as a metric (via an oracle), or given by a smooth kernel function in a Euclidean space. For both of these settings, we show that for any $\epsilon > 0$, one can compute an explicit $(1 + \epsilon)$-approximate spectral approximation of the geometric graph with $\tilde{O}(n/\epsilon^2)$ edges in $\tilde{O}(n/\epsilon^2)$ randomized time. Some of these algorithms are extremely simple. Composed with nearly linear time graph algorithms, this allows for a broad class of applications on geometric graphs with running times proportional to the number of points.

*Purdue University. krq@purdue.edu. https://kentquanrud.com.

1 Introduction

Many geometric problems can be formulated graphically as follows. The input consists of n points V in some metric space, and one considers an *implicit* complete undirected graph over V where the weight of an edge is a function of the distance between its endpoints in the underlying geometry. A simple question such as computing the average degree in this graph is both useful for many applications, and difficult without inspecting every edge. Usually, in combinatorial settings where the graph is given explicitly, a running time proportional to the number of edges equates to a running time proportional to the input size, and is satisfactory. Geometric graphs are implicit, and the *explicit size of the graph*, proportional to n^2, is *quadratic in the size of the input*, proportional to n. This simple obstacle makes many basic problems nontrivial. To circumvent this n^2-time barrier, algorithms are designed to take advantage of the underlying geometry. In many cases these algorithms are *approximate* and *randomized*. For example, one can efficiently approximate the average degree by random sampling approaches informed by the geometry, in multiple contexts discussed below. In this work, we consider approximations for the broad class of optimization problems defined on these implicit graphs, and devise general strategies to obtain running times proportional to n, rather than n^2.

This work is focused on two broad classes of implicit geometric graphs, where the edges weights either form a *metric*, or where the edge weights are given by a *kernel* over a Euclidean space.

1.1 Metrics

Arguably the simplest and most ubiquitous geometric setting is *metric spaces*. The input consists of n points V and a metric $d : V \times V \to \mathbb{R}_{\geq 0}$ given implicitly via an oracle. We interpret the input as a complete graph $G_d = (V, E)$ with edge weights given by d. The goal is to solve combinatorial problems on G_d with algorithms that either (a) run in time substantially smaller than the explicit size of the metric – $O(n^2)$ – or otherwise (b) query substantially less than all $\binom{n}{2}$ lengths in the metric. Such algorithms are called "sublinear time algorithms" in the literature – "sublinear" is meant with regards to the explicit size of the metric. This model was initiated by the seminal work of Indyk [28, 29], which showed the first sublinear bounds for approximating a variety of basic problems (such as average degree, 1-median, bicriteria approximations for k-median, 2-clustering, and maximum cut) as well as lower bounds for problems that do not permit sublinear algorithms. We refer to the excellent survey by Czumaj and Sohler [18] on sublinear algorithms in metrics, particularly in the context of property testing. More recently, Chechik, Cohen, and Kaplan [16] gave randomized algorithms that, among other results, could $(1 \pm \epsilon)$-approximate the sum of all distances in a metric with high probability in $O\big(n + \epsilon^{-2} \log(n)\big)$ queries, and $(1 \pm \epsilon)$-approximate the degree of all of the vertices with $O\big(n \log(n)/\epsilon^2\big)$ queries, improving previous work from [29]. Their algorithms are also efficient in the implicit setting of the shortest path metric in a graph (where one does not have a point to point oracle, but one can obtain single-source distances in nearly linear time). Another recent work by Esfandiari and Mitzenmacher [20] is based on importance sampling edges. Their results include a nearly-linear time $1/2 - \epsilon$ approximation for densest subgraph and a sublinear oracle complexity (though not sublinear running time) for maximum k-hypermatching. Another result improves ϵ-dependence in the oracle complexity for maximum cut as compared to [28] (although not the running time). [20] also showed a lower bound of $\Omega(n)$ queries for the problems of approximating the sum of distances, the densest subgraph, and the maximum cut. Note that the lower bound for estimating the sum of distances is tight with the algorithm of [16]. We also refer the reader to [20] for additional references, which are still up to date.

1.2 Kernels

We consider the analysis of graphs over n points in moderate to high-dimensional Euclidean spaces $\mathbb{R}^d$, where the weight of an edge between two points is given implicitly by a **kernel function** $\kappa : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}_{\geq 0}$

that typically decay with the Euclidean distance between the two points. This definition might appear derived compared to metrics, but there are also a large number of applications using kernels. In physics, kernel-weighted graphs arise in n-body systems where each pair of bodies x and y have some kind of force between them (e.g., gravitational pull) that decays with the Euclidean distance between x and y. Kernels also arise in statistical analysis, where (for example) $\kappa(x, y)$ may reflect the likelihood of one data point given another (under some distributional assumptions). Throughout machine learning kernels are used to measure similarity between points.

A primary computational task related to kernels is **kernel density estimation**. In density estimation, given a set of points V and a query point q, the goal is to estimate either the total or the average of kernel valuations $\kappa(q, v)$ over all $v \in V$. In n-body systems in physics, this may measure the potential at a point. In statistics, this is a reasonable estimate of the density of a distribution where the set V reflects empirical observations. Density estimation is an important primitive for many applications in machine learning [21, 22, 38, 40, 44].

The scientific computing community has been very successful in designing algorithms for kernel density estimation problems in low-dimensional spaces (e.g., $d = 2$ or 3) as arise in physics. The most famous of these results is the celebrated fast multipole method by Greengard and Rokhlin Jr. [24]. The fast multipole method has been applied to a number of related problems and we refer to [9, 12, 23] for further background.

For many applications in statistical analysis and machine learning, the dimension d is large. The fast multipole method relies on geometric data structures that scale poorly with the dimension. More recent work has tried to handle high-dimensional point sets with sampling based approaches. Holmes, Gray, and Isbell Jr. [27] first gave the following value-dependent bound for density estimation. For $\mu \in (0, 1)$, to decide if the density is less than μ up to a $(1 \pm \epsilon)$-multiplicative error, [27] showed that it sufficed to evaluate the kernel at $1/\epsilon^2\mu$ (randomly selected) points. The number of kernel evaluations was reduced by Charikar and Siminelakis [14] to roughly $1/\epsilon^2\sqrt{\mu}$ kernel evaluations, using sophisticated techniques such as locality-sensitive hashing [30] to generate appropriate nonuniform sampling probabilities. Followup work in [41] considered some of the more practical aspects of [14] and, with some algorithmic enhancements and refined analysis, was able to demonstrate empirical improvements that support the theory. Additional enhancements to the approach of [14] are developed in [6]. Backurs, Indyk, and Schmidt [5] showed that some dependence on μ is necessary under the Strong Exponential Time Hypothesis. In pursuit of more efficient algorithms with bounds that are independent of μ, Backurs, Charikar, Indyk, and Siminelakis [4] considered a restricted class of **smooth kernels** that decay polynomially in the distance. This class is broad enough to include many of the more popular kernel functions. For this class, [4] gave density estimation algorithms that require (roughly) just $\log(n)/\epsilon^2$ kernel evaluations. Their techniques also use or are inspired by locality sensitive hashing. [15] extends hashing based techniques to additional kernels such as log-convex kernel functions. These groundbreaking algorithms for kernel density estimation have inspired further applications related to kernels that either use density estimation directly or are otherwise inspired by the underlying hashing techniques.

In this work, we treat the input points as vertices of a graph with edges weighted by a smooth kernel κ. We call this a **kernel graph**. We believe it would be very fruitful to solve graph partitioning, embedding, or clustering problems on kernel graphs. To place the previous literature in this context, observe that the kernel density at a point is the weighted degree of that point in the kernel graph.

While the graphical perspective towards a kernel κ is natural and expressive, it is also computationally expensive. The basic obstruction in modeling a kernel as a graph is the simple difficulty of constructing the graph. The graph is a clique and each pair of points requires evaluating κ to determine the weight of the edge between them. Meanwhile, all of the algorithmic effort discussed above was entirely to avoid such an all-to-all kernel computation. If one cannot build the kernel graph, then the idea of running sophisticated graph algorithms on kernels is moot.

1.3 Algorithms and results

In this work, we do not try to construct metric graphs or kernel graphs exactly. Instead, we ask it is possible to *explicitly construct a sparse approximation of these geometric graphs in nearly linear time*. We want the running time to be comparable to the above literature (for more specific problems), and in particular, roughly proportional to n. The graph should be constructed explicitly so that existing graph algorithms can be applied directly. The graph should also be sparse to decrease the input size to algorithms downstream. (Sparsity is also necessitated by the running time requirement.) To define our notion of approximation, we take a spectral perspective on kernel weighted graphs. We seek to produce a graph whose *Laplacian* quadratic form preserves the Laplacian of the geometric graph up to a controlled relative error. The Laplacian encodes many aspects of the graph (some of which are listed later), so many graph algorithms applied to the spectral approximation beget approximations w/r/t the original geometric graph – without ever building the geometric graph explicitly.

We depart conceptually from most previous work in the sense that we are principally concerned with preserving the structure of the entire graph. Previous work tends to be more problem-specific and interested in estimating a precise set of quantities. The narrowed scope in previous work offers more structure to leverage and limits the potential combinations one must account for in the design and analysis of these algorithms. The relatively problem independent approach here tends to lead to observations about the *structure* of some of these geometric graphs in general, and the output, a spectral sparsifier, can be used in many different ways. The guiding hope is that an efficient spectral approximation to geometric graphs might enable a vast and well-developed catalog of graph algorithms to be applied to geometry problems, and open up a new combinatorial perspective on them. An inspiring example of a previous result in this spirit is the *well-separated pair decomposition* of Callahan and Kosaraju [13] for low-dimensional Euclidean spaces, which lead to spanners (i.e., auxiliary graphs that preserve all distances) and many other results in computational geometry.

1.3.1 Spectral sparsification of metrics

The first result in this work is an easily implementable algorithm that produces a sparse spectral approximation of a metric in nearly linear time and queries.

Theorem 1.1. *Given a metric d over n points accessed via an oracle, there is a randomized algorithm that with high probability, returns a weighted graph with $O\left(n \log(n)/\epsilon^2\right)$ edges that is a $(1 \pm \epsilon)$-spectral approximation of the weighted graph induced by d in $O\left(n \log(n)\left(\log(n) + Q\right)\right)$ randomized time, where Q denotes an oracle query to the metric d.*

The key ingredient in the above theorem is the following *uniform upper bound* on the *effective resistance* of all the edges in a metric. The effective resistance between two vertices in an undirected graph is introduced in greater detail in Section 2. It is the quantity obtained by interpreting the graph as an electrical network, and measuring the resistance of this network as if it were a single resistor between the two vertices. We refer to the effective resistance of the endpoints of an edge e as simply the effective resistance of e.

Theorem 1.2. *Let $G = (V, E)$ be a complete undirected graph with edge weights given by a metric d. Let $D = \sum_{e \in E} d(e)$ be the sum of distances over all pairs of vertices. For all edges $e \in E$,*

$$(\textit{effective resistance of } e) \leq \frac{cn}{D}$$

for some universal constant $c > 0$.

Spielman and Srivastava [43] showed that in general undirected graphs, importance sampling edges in proportion to their individual weights times their effective resistances produces a random graph whose

Laplacian is strongly concentrated around the original graph. Usually some effort is required to compute the effective resistances, which for geometric graphs would lead to quadratic running times. Theorem 3.2 instead gives a universal upper bound that is still strong enough to produce sparse graphs within the framework of [43]. Theorem 1.2 says that in a metric, one only needs to importance sample edges in proportion to their lengths to produce a spectral approximation.

To complete the algorithm in Theorem 1.1, it remains to crudely estimate the length of each edge with very few queries. Here one could use a subroutine in [20], but we propose an alternative that is simpler and improves the query complexity by a logarithmic factor. All put together, the total algorithm for Theorem 1.1 is extremely simple to implement. The algorithm non-uniformly samples $O(n\log(n)/\epsilon^2)$ edges with repetition in proportion to crude estimates of the edge lengths. The sampled edges are added to the graph with weights scaled (roughly) in inverse proportion to the edge's probability of being sampled. The crude estimates are obtained by first uniformly sampling $O(n\log n)$ edges. For each edge e, we estimate their length by taking the sum of lengths of the uniformly sampled edges incident to e, divided by $\log(n)$.

While Theorem 1.1 is based on the abstract oracle model, the results are also interesting for the explicit setting of Euclidean distance in $\mathbb{R}^d$. An important tool in this setting is random projection [31], with which one can randomly project down to $O(\log(n)/\epsilon^2)$ dimensions and preserve all pairwise distances up to a $(1\pm\epsilon)$-multiplicative factor. For example, Barhum, Goldreich, and Shraibman [7] observed that to estimate the average Euclidean distance in $\mathbb{R}^d$ with constant probability of success, it suffices to project onto $O(1/\epsilon^2)$ dimensions. For the spectral sparsifier presented here, when the dimension d is greater than $\log(n)/\epsilon^2$, one can use the *sparse* Johnson-Lindenstrauss transform [1, 19, 32, 33] to reduce the running time slightly from $O(n(d+\log(n))\log(n)/\epsilon^2)$ to $O\left(nd\log(n)/\epsilon + n\log^2(n)/\epsilon^4\right)$. We found it surprising that the spectral sparsifier presented here is already fast enough that the classical construction [31] (with a dense random projection matrix) does not actually improve the running time. With regards to some of the problem-specific lower bounds in [20], Theorem 1.1 reduces the oracle complexity of approximating maximum cut by a $O(\log n)$ factor from $O\left(n\log^2 n\right)$ to $O(n\log n)$, reducing the gap to one logarithmic factor from the $\Omega(n)$ lower bound.

1.3.2 Spectral sparsification of kernels

In this section, we give an overview of the algorithmic results for kernel graphs. To emphasize the high-level ideas, we give simplified bounds here, deferring the detailed versions of the theorems to later sections. The algorithms apply to *smooth kernels*, defined in Section 4.1, and essentially the same as the notion defined by [4]. Smoothness is parametrized by a value $t > 0$. Loosely speaking, a kernel is t-smooth if $\kappa(x, y)$ decays at a rate of roughly $1/\|x-y\|^t$.

Theorem 1.3. *Let $\kappa : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}_{\geq 0}$ be a smooth kernel, and consider a kernel graph G_κ over n points in $\mathbb{R}^d$. Let Q denote the time to query the kernel. With high probability, one can construct a spectral sparsifier of G_κ with $\tilde{O}(n/\epsilon^2)$ weighted edges in $\tilde{O}(n(d+Q)/\epsilon^2)$ randomized time.* [1]

The algorithm underlying Theorem 1.3 is based on importance sampling, where we first assign an "importance" to each edge, and then sample $\tilde{O}(n/\epsilon^2)$ edges in proportion to their importance. Each time an edge is sampled, we add the edge to the graph with the edge weight scaled up in inverse proportion to the sampling probability (so that the randomized weight of each edge is unbiased). From the perspective of spectral graph theory, the "importance" of an edge is given by its effective resistance (see Section 2). To

[1]Here "$\tilde{O}(\cdots)$" hides logarithmic dependencies on the number of points, the multiplicative spread of the distances between them, an exponential dependence on t, and a linear dependence on a second smoothness parameter β. For $t = 1$, there is no dependence on spread. The exponential dependence on t is necessary by standard complexity assumptions [4]. "High probability" signifies a probability of error $\leq 1/\text{poly}(n)$.

apply this technique to kernel-weighted graphs, we need to obtain overestimates on the product of the kernel and the effective resistance between every pair of vertices. To obtain $O(n^2)$ such values in time proportional to n, we need to obtain these overestimates implicitly. The number of edges sampled is proportional to the sum of overestimates. To keep the output sparse, we want to keep the overestimates as tight as possible in the aggregate.

The algorithm to overestimate all effective resistances for $t = 1$ is remarkably simple. Let $a \sim \mathcal{N}^d$ be a random d-dimensional vector where each coordinate is independently distributed as a standard Gaussian, and consider the random line embedding $v \mapsto \langle a, v \rangle$. Based on this embedding, we can assign a "rank" to each vertex enumerating the vertices $v \in V$ in increasing order of $\langle a, v \rangle$. For each edge $e = \{u, v\}$, consider the difference in ranks of its endpoints, $|\text{rank}(x) - \text{rank}(y)|$. Intuitively, if $\kappa(x, y)$ is relatively large (relative to other edges incident to x or y), then $\|x - y\|$ is relatively small, and we would expect $|\text{rank}(x) - \text{rank}(y)|$ to be small. Conversely, for large $\kappa(x, y)$, we would expect $|\text{rank}(x) - \text{rank}(y)|$ to be big. It is thus reasonable to sample each edge $e = \{x, y\}$ with probability *inversely proportional* to $|\text{rank}(x) - \text{rank}(y)|$. One can justify this heuristic formally by proving that with constant probability, $1/|\text{rank}(x) - \text{rank}(y)|$ is a constant factor upper bound on the product of the kernel of and the effective resistance between x and y. (The proof of this claim is not as simple as the algorithm.) Consequently, one can take the average of $1/|\text{rank}(x) - \text{rank}(y)|$ over a few independent trials and scale up by a constant to obtain an upper bound on the kernel times the effective resistance with high probability. It is easy to see that the sum of $1/|\text{rank}(x) - \text{rank}(y)|$ over all $\{x, y\} \in E$ is $O(n \log n)$, and it is easy to sample edges in proportion to $1/|\text{rank}(x) - \text{rank}(y)|$ as well. Thus one obtains a spectral sparsifier in nearly linear time.

The algorithm for larger t is similar, except we now randomly project onto t dimensions, and enlist the help of some simple data structures. The data structure is a quadtree[2] over the projected point sets in $\mathbb{R}^t$. After building quadtrees over independent projections of the vertices, the nonuniform sampling is conducted implicitly as follows. We choose a quadtree and a vertex uniformly at random. Out of the canonical cells containing x in the quadtree, we choose one cell uniformly at random. We examine this cell along with its neighboring cells at the same level of the quadtree, and sample one vertex y from these cells uniformly at random. We then add the edge $\{x, y\}$ with its weight scaled up appropriately. The intuition for this scheme is similar to the simpler setting with $t = 1$. Relatively large kernels $\kappa(x, y)$ belong to pairs x and y that are close together in $\mathbb{R}^d$, and are more likely to be adjacent at smaller cells of the quadtree. The number of other vertices nearby at the same resolution give (a weaker, higher-dimensional analogue of) the "difference in rank" between x and y. One can show that with fixed probability (depending on t), a single quadtree (implicitly) obtains a overestimate (up to constants) of the kernel times the effective resistance between x and y. Building a few quadtrees over independent projections ensure we overestimate each effective resistance with high probability.

Density estimation. Recall the high-dimensional density estimation problem addressed by previous work [4, 14]. Here we want to preprocess a set of points V, such that given a query point q, we can estimate the sum of kernel evaluations between q and V, $\sum_{v \in V} \kappa(q, v)$, with much fewer than $|V|$ kernel evaluations. The spectral sparsifier in Theorem 1.3 naturally implies an algorithm for kernel density estimation. Given a query point q, consider the kernel-weighted graph over the augmented vertex set $V + q$. The density of q w/r/t V is precisely the weighted degree of q in this graph. Suppose we build a spectral sparsifier over $V + q$. The spectral sparsifier preserves all of the cuts, and in particular preserves the cut induced by the singleton set $\{q\}$ – which is the degree of q, which is the density of V at point q. Of course, the kernel density at q does not take into consideration any of the edges between points in V, so the effort to sparsify these internal edges can be omitted. This leads to the following theorem, proven in Section 4.6.

[2]Here enters a logarithmic dependence on the spread.

Theorem 1.4. *In $\tilde{O}(n)$ preprocessing time and space, one can compute $(1 \pm \epsilon)$-multiplicative approximations to kernel densities in $\tilde{O}(1/\epsilon^2)$ time.*

While the lower order terms are omitted here, the detailed bounds of Theorem 1.4 (given in Theorem 4.18 in Section 4.6) actually improve the state of the art in [4]. For obtaining $(1 \pm \epsilon)$-approximations with high probability, the preprocessing time and space are both reduced by a $O(1/\epsilon^2)$-multiplicative factor. The query times are similar.

1.4 Applications

The potential applications of the spectral sparsifiers are many, given the plethora of applications of spectral graph theory [34, 42, 45]. For example, Laplacians encode the values of all cuts. Hence a spectral approximation implies that all cuts are approximately preserved. Any cut-based optimization algorithm – such as for max flow, sparsest cut, or graph partitioning – can be applied to the sparse spectral approximation to give an approximation on the original, dense, geometric graph. We note that recent developments have lead to nearly linear time graph algorithms for many of these problems, which leads to graph algorithms for metric and kernel weighted graphs with running times proportional to the number of vertices, even though these graphs are dense.

The Laplacian is also important for spectral embeddings. Here one computes one or a few of the top or bottom eigenvectors of the Laplacian. These eigenvectors give a low-dimensional embedding of the vertices (one coordinate per eigenvector). For applications of metric graphs, one would probably use the eigenvectors corresponding to the largest eigenvalues, where points far away in the metric are also spread out by the eigenvectors. For many applications of kernel weighted graphs, where the kernel is used to measure similarity, one might use the eigenvectors corresponding to the smallest eigenvalues instead. Then pairs of vertices with large kernels / greater similarity are placed closer together. The embedding is useful for clustering and visualization, among other applications [10, 37]. Rather than compute a spectral embedding over the complete graph with all pairs of kernel evaluations, one can first apply Theorem 1.3 to compute a sparse graph with approximately the same spectrum, and embed this graph instead. In general, the spectral sparsifiers reduce the spectral analysis of dense geometric graphs to sparse graphs in nearly linear time, enabling a broad class of spectral analyses to run in time proportional to the number of vertices rather than the number of pairs of vertices.

1.5 Additional related work

There is an extensive literature on how to solve (or approximate) optimization problems on point sets in *low* dimensions, often employing some sort of grid or quadtree over the low-dimensional space. We refer to the monograph by Har-Peled [25] for a broad overview of many such problems and techniques. We highlight a few examples. *Spanners* are weighted auxiliary graphs, generally sparse, whose shortest path metrics approximate an underlying metric. As a sparse graph approximating certain properties of the geometric graph, spanners are similar in spirit to the spectral sparsifiers here. One can construct spanners over Euclidean point sets with roughly n edges (for small d) via the well-separated pair decomposition [13] (among other approaches [39, 47]). We note that [13] was also motivated by n-body problems and that well-separated pair decompositions have many other applications; see [25, Chapter 3]. Well-separated pair decompositions can also be constructed from quad trees [26]. The celebrated approximation schemes for Euclidean TSP [3, 36] (and other network design problems) are also based on low-dimensional grids. Here hierarchical cells are used to discretize the solution space and impose a hierarchy of subproblems suitable to dynamic programming. The results in this work do not require the input points to be in low-dimension. For sparsifying kernels, we do take advantage of the *random projections* of the point sets lying in low-dimension, and then benefit from the general ideas and techniques developed in low-dimensional settings.

For kernels in high dimensions, the general strategy of randomly projecting the point set into a lower-dimension space to generate non-uniform sampling probabilities starts with [14] and recurs throughout the recent developments on this problem [4, 6, 15, 41]. The ideas are inspired by (or otherwise use directly) locality sensitive hashing, which was introduced by [30] and is an important technique for approximating high-dimensional problems.

1.6 Independent work

Finally, we would like to mention independent work by Alman, Chu, Schild, and Song [2], which is strongly related to this work. They derive some similar algorithmic results for kernels among other functions of Euclidean distances, and also give hardness results. We defer a more detailed discussion to a future version of this paper, and recommend [2] to the interested reader.

1.7 Organization

The rest of this work is organized as follows. In Section 2, we given preliminaries on spectral graph theory useful to both the metric and kernel sparsification algorithms. In Section 3, we define and analyze the spectral sparsification algorithm for metrics. In Section 4, we define and analyze the spectral sparsification algorithm for kernels. Section 4.6 contains the kernel density estimation algorithm derived form the spectral sparsifier. Section 3 and Section 4 can be read separately. They are unified in motivation by the spectral sparsification framework, but the technical challenges and techniques to overcome them in each setting are very different.

2 Preliminaries

In this section we give preliminary background on spectral graph theory that is used for both metrics and kernels. Additional background can be found in [17, 42].

Given an undirected graph $G = (V, E)$ with nonnegative edge weights $w : E \longrightarrow \mathbb{R}_{\geq 0}$, the **Laplacian of *G***, denoted L_G, is the positive semidefinite matrix $L_G \in \mathbb{R}^{V \times V}$ defined by the quadratic form

$$\langle x, Lx \rangle = \sum_{e=\{u,v\} \in E} w(e)(x_u - x_v)^2$$

Given two graphs G and H over the same vertex set, H is said to be a **$(1 \pm \epsilon)$-spectral approximation** of G if

$$(1-\epsilon)L_G \preceq L_H \preceq (1+\epsilon)L_G.$$

That is, for any vector $x \in \mathbb{R}^V$, we have

$$(1-\epsilon)\langle x, L_G x \rangle \leq \langle x, L_H x \rangle \leq (1+\epsilon)\langle x, L_G x \rangle.$$

The goal of spectral sparsification is to take an undirected graph G and compute a $(1 \pm \epsilon)$-spectral approximation H with as few edges as possible. Spielman and Srivastava [43] showed that every graph on n vertices has a $(1 \pm \epsilon)$-spectral approximation with $O\big(n \log(n)/\epsilon^2\big)$ edges, which was then reduced further to $O\big(n/\epsilon^2\big)$ edges [8]. Moreover, these sparsifiers can be computed in nearly linear time [35, 43].

Given a graph G with Laplacian L_G, the **effective resistance** of an edge $e = \{u, v\}$, denoted (eff. resist. e), is defined by

$$\frac{1}{(\text{eff. resist. } e)} = \min\big\{\langle x, Lx \rangle : x \in \mathbb{R}^V,\, x_u = 0,\, x_v = 1\big\}.$$

To sparsify kernel weighted graphs, following [43], we sample edges with probabilities proportional to their effective resistance. For the sake of efficiency, we prefer to sample the edges with replacement, rather than sample each edge separately as in [43]. In particular, we will use the following lemma.

Lemma 2.1. *Let $G = (V, E)$ be a weighted undirected graph with n vertices, and for each $e \in E$, let*

$$p_e \geq (\textit{weight of } e)(\textit{eff. resist. } e). \tag{1}$$

Let $m = \alpha \sum_e p_e$ for $\alpha = O(\log(n)/\epsilon^2)$. Consider the randomized weighted graph H where we sample m edges with replacement in proportion to $\{p_e, e \in E\}$, where for each sampled edge e, we add e to H with weight scaled up by $1/\alpha p_e$. Then with high probability, H is a $(1 \pm \epsilon)$-spectral sparsifier.

Proof. We include this proof for the sake of completeness. For each edge e, let L_e denote the (weighted) Laplacian of the edge e. Let $L_G = \sum_e L_e$ denote the Laplacian of G. Note that for each edge e, $p_e^{-1} L_e \preceq L_G$. Indeed, for any vector $x \in \mathbb{R}^V$, we have

$$\left\langle x, \left(p_e^{-1} L_e\right) x \right\rangle = \frac{(\text{weight of } e)}{p_e} (x_u - x_v)^2 \overset{\text{(a)}}{\leq} \frac{1}{(\text{eff. resist. } e)} (x_u - x_v)^2 \overset{\text{(b)}}{\leq} \langle x, Lx \rangle. \tag{2}$$

Here (a) is by choice of p_e per inequality (1), and (b) is by definition of effective resistance. For $i = 1, \ldots, m$, let e_i be the ith sampled edge, and let L_i denote the Laplacian of this reweighted edge. Let $L_H = \sum_i L_i$ denote the Laplacian of H. L_H is a random matrix.

Let $P = \sum_{e \in E} p_e$. For each $i \in [m]$, we have $L_i = L_e / \alpha p_e$ with probability p_e / P for each edge e. In expectation, we have

$$\mathrm{E}[L_i] = \sum_{e \in E} \frac{p_e}{P} \cdot \left(\frac{1}{\alpha p_e}\right) L_e = \frac{1}{\alpha P} L_G = \frac{1}{m} L_G.$$

Summing over all i, the expectation of L_H is $\mathrm{E}[L_H] = \sum_{i=1}^m \mathrm{E}[L_i] = L_G$. We also have

$$L_i = \left(\frac{1}{\alpha p_{e_i}}\right) L_{e_i} \overset{\text{(c)}}{\preceq} \frac{\epsilon^2}{c \log n} L_G$$

for some sufficiently large constant c deterministically. Here (c) is by inequality (2). The claim now follows from matrix concentration (see, e.g., [42, 46]). ■

Of course, one can compose Lemma 2.1 with sparsifiers in [8, 35] to reduce the number of edges in H to $O(n/\epsilon^2)$.

3 Metrics

In this section, we design and analyze algorithms for computing spectral sparsifiers of metrics. We first give a high level overview of the results.

Let $d : V \times V \to \mathbb{R}_{\geq 0}$ be a metric on a set of n points V. We interpret d as a complete undirected graph on V, denoted $G_d = (V, E)$, with edge weights given by the metric d. For an edge $e = \{u, v\} \in E$, we let $d(e)$ denote the distance between its endpoints in the metric, $d(u, v)$. Let $D = \sum_{e \in E} d(e)$ denote the sum of distances over all pairs of points.

The algorithms in this work are based on importance sampling. Generally speaking, importance sampling reduces the variance in uniform sampling by sampling "more important" objects with high probability. For metrics, we sample edges in proportion to their edge lengths. It remains to show (a) how to estimate the

```
apx-metric-importance($d : V \times V \longrightarrow \mathbb{R}_{>0}$)

1. let $F$ sample each edge with probability $p = 8\log(n)/(n-1)$

2. for each vertex $u$

   A. $Y_u \leftarrow \frac{1}{\log n}$(sum of distances of sampled edges incident to $u$)

// for each edge $e = \{u, v\}$ denote $Y_e = Y_u + Y_v$

3. return $Y$.
```

Figure 1: A subroutine for overestimating the length of every edge with $O(n \log n)$ total queries and constant overhead in the total distance (see Theorem 3.1).

importance/length of each edge with $O(n \log n)$ total queries, and more importantly (b), that the edge lengths are a good measure of importance in the spectral sense.

The first step (a) assigns overestimates on the length of every edge, while keeping the total sum of overestimates comparable to the total sum of distances. The algorithm, `apx-metric-importance`, is very simple. We uniformly sample $O(n \log n)$ edges in expectation. For each edge e, we sum the distances of all sampled edges incident to e, and scale down by $\log(n)$. This will overestimate each e with high probability. Because each sampled edge contributes a $\log(n)$-fraction of its own distance to all the incident edges, the sum of estimates is at most a constant factor greater than the full sum of distances with high probability. The formal bounds are as follows. The formal proof (which is not much longer than this paragraph) is given in Section 3.1.

Theorem 3.1. *With high probability,* `apx-metric-importance` *makes $O(n \log n)$ queries and returns values Y_e for $e \in E$ satisfying the following properties.*

(i) For all $e \in E$, $Y_e \geq d(e)$.

(ii) $\sum_e Y_e \leq O(1)D$.

The next part of the algorithm samples edges with replacement in proportion to the estimates Y_e. More precisely, we apply the sample procedure from Lemma 2.1, where each p_e is taken in proportion to Y_e. To show that this process is well-concentrated, we have to analyze the effective resistances of the edges. We obtain the following uniform bound on the effective resistance of all edges in a metric. The proof of the following theorem is given in Section 3.2.

Theorem 3.2. *Let $G = (V, E)$ be a complete undirected graph with edge weights given by a metric d. Let $D = \sum_{e \in E} d(e)$ be the sum of distances over all pairs of vertices. For all edges $e \in E$,*

$$(\textit{effective resistance of } e) \leq \frac{cn}{D}$$

for some universal constant $c > 0$.

The overall algorithm, given in Figure 2, is as follows. We obtain estimates $Y_e \geq d(e)$ for each edge (by Theorem 3.1). Let

$$p_e = O(1) Y_e \frac{n}{\sum_e Y_e}.$$

```
metric-spectral-sparsifier($d : V \times V \to \mathbb{R}_{\geq 0}$, $\epsilon$)
1. $\alpha \leftarrow \Omega\left(\epsilon^2/\log n\right)$ where $n = |V|$, $w \leftarrow \mathbb{0}^E$
2. $Y \leftarrow$ apx-metric-importance($d$)
3. apply Lemma 2.1 and importance sample $O\left(n\log(n)/\epsilon^2\right)$ edges with replacement w/r/t $p_e = \frac{cY_e n}{\sum_e Y_e}$
   for $e \in E$, for some universal constant $c > 0$
```

Figure 2: A spectral sparsifier for metrics taking $O\left(n\log(n)/\epsilon^2\right)$ time and queries to produce a $(1 \pm \epsilon)$-spectral approximation with high probability (see Theorem 3.3).

Then with high probability, for all $e \in E$, we have

$$p_e = O(1)Y_e \frac{n}{\sum_e Y_e} \overset{\text{(d)}}{\geq} O(1)d(e)\frac{n}{D} \overset{\text{(e)}}{\geq} d(e)(\text{eff. resist. } e).$$

where (d) is by Theorem 3.1 and (e) is by Theorem 3.2. By Lemma 2.1, importance sampling $O\left(\frac{\log(n)}{\epsilon^2}\sum_e p_e\right)$ edges with replacement in proportion to p_e returns a $(1 \pm \epsilon)$-spectral sparsifier with high probability. We have $\sum_e p_e = O(n)$, and sampling in proportion to p_e is the same as sampling in proportion to Y_e and easy to generate in $O(\log n)$ time per edge[3]. Thus we have the following.

Theorem 3.3. *With high probability,* metric-spectral-sparsifier *returns a weighted graph with $O\left(n\log(n)/\epsilon^2\right)$ edges that is a $(1 \pm \epsilon)$-spectral approximation of d in $O\left(n\log(n)(\log(n) + Q)\right)$ randomized time, where Q denotes an oracle query to the metric.*

Besides metrics. Like [20], we observe that these techniques extend to nonnegative symmetric functions $d : V \times V \to \mathbb{R}_{\geq 0}$ that are not metrics, but still similar to metrics. For $\alpha \geq 1$, we say that d is an **α-approximate metric** if it satisfies the triangle inequality up to any α-multiplicative factor in the sense that $d(a,b) \leq \alpha(d(a,c) + d(c,b))$ for any three points a, b, c. ([20] calls this an α^{-1}-metric.) One can retrace the arguments for metrics and obtain similar results with a multiplicative increase of α in the bounds.

3.1 Metric importance of edges

In this section, we analyze the subroutine apx-metric-importance. The goal of apx-metric-importance is to assign an overestimate on the length of each edge (claim (i)), while keeping the sum of overestimated length as close to the true sum of lengths as possible (claim (ii)).

Lemma 3.4. *For all $e \in E$, we have $Y_e < d(e)$ with probability of error $\leq 1/\text{poly}(n)$.*

Proof. Let $p = \Theta(\log(n)/n)$ be the probability with which each edge is sampled. Let $e = \{a, b\}$. For any third point $c \in V \setminus e$, we have

$$\max\{d(e), d(a,c)\} + \max\{d(e), d(c,b)\} \geq d(a,c) + d(c,b) \overset{\text{(a)}}{\geq} d(e)$$

by (a) the triangle inequality. Even if we truncate all incident edges to contribute at most $d(e)$ to Y_e, the expected sum is $\geq pnd(e) = \Omega(\log(n)d(e))$. W/r/t the truncated distances, each edge contributes at most a $\Theta(\log n)$-fraction to the expected sum. The claim now follows from standard concentration bounds. ■

[3]Here we point out that sampling an edge in proportion to Y_e is the same as sampling two endpoints in proportion to the vertex sums Y_u.

Lemma 3.5. *$\sum_e Y_e \leq O(1)D$ with high probability.*

Proof. Each sampled edge e contributes at most $nd(e)/\log(n)$ to the total sum of distances, and $\sum_h d(h) \geq nd(e)$ for all e by the triangle inequality. The claim follows from standard concentration bounds. ∎

3.2 Spectral importance of edges

In this section, we prove Theorem 3.2, which upper bounds the effective resistance of an edge in a metric. We first restate Theorem 3.2 for convenience.

Theorem 3.2. *Let $G = (V, E)$ be a complete undirected graph with edge weights given by a metric d. Let $D = \sum_{e \in E} d(e)$ be the sum of distances over all pairs of vertices. For all edges $e \in E$,*

$$(\textit{effective resistance of } e) \leq \frac{cn}{D}$$

for some universal constant $c > 0$.

Proof. Fix a vector $x \in \mathbb{R}^V$. For an edge $f = \{c, d\}$, we denote

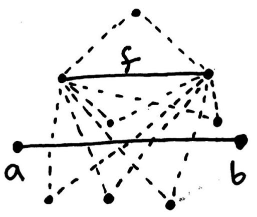
$$x^2(f) = x^2(c,d) \overset{\text{def}}{=} (x_c - x_d)^2.$$

We want to show that

$$\sum_{f \in E} d(f)x^2(f) \geq c\frac{Dx^2(e)}{n}$$

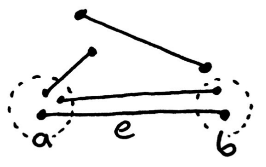
for some universal constant $c > 0$.

We first give a high-level overview of the argument. For the sake of conversation, suppose that $x^2(e) = 1$. We want to show that the total amount of potential, over all edges, is roughly the total distance in the metric divided by n. The analysis partitions E and argues that in each part we have the desired inequality. The parts are first generated by repeatedly selecting the longest "far" edge $f \neq e$ in the graph (where "far" will be defined shortly), and select the set F of all edges incident to $f \setminus e$.

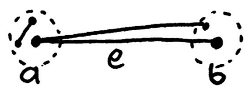


Each such f generates one part and then the points in $f \setminus e$ (and all incident edges F) are removed from the metric. When there are no "far" edges left, then we analyze all of the remaining "close" edges (which includes e) as the final part of the decomposition.

The edges are classified as either "close" or "far" from e as follows. First, let B_a be the ball of radius $d(e)/4$ centered at a, and let B_b be the ball of radius $d(e)/4$ centered at b. We call an edge f **far** if $f \setminus e$ is not contained in B_a and not contained in B_b. Otherwise we call f **close**.



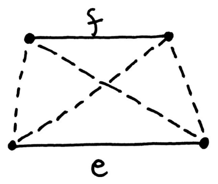
Far edges.



Close edges.

Far edges and close edges have different properties that make them useful for the decomposition. The proofs of these properties are detailed and obscure some of the high-level ideas of the proof. For the moment, we state the main observations about far and close edges required to prove the overall theorem, and return to their proofs in later sections.

For far edges, we state two lemma's. The proofs for these claims are given in Section 3.2.1 below. The first lemma relates the potential of f and its associated edges between f and e to the distance between the endpoints of f.



Lemma 3.6. *Let f be a far edge, and let F be the set of edges with both endpoints in $f \cup e$. Then*

$$\sum_{h \in F, h \neq e} d(h)x^2(h) \geq c \max\{d(e), d(f)\} x^2(e)$$

for some universal constant $c > 0$.

The second lemma observes that as long as f is the largest far edge from e, its edge length compares favorably to the length of any edge incident to $f \setminus e$.

Lemma 3.7. *Let f be the largest edge such that $f \setminus e$ is not contained in B_a and not contained in B_b. Let h be any edge incident to f and not incident to e. Then*

$$d(h) \leq c \max\{d(f), d(e)\}.$$

for some universal constant $c > 0$.

When all the edges are close, then the following lemma states that the desired inequality already holds. Lemma 3.8 is proven below in Section 3.2.2.

Lemma 3.8. *Suppose every edge is close. Then*

$$\sum_{f \in E} d(f)x^2(f) \geq c \frac{Dx^2(e)}{n}.$$

for some universal constant $c > 0$.

We now prove the theorem assuming Lemma 3.6, Lemma 3.7, and Lemma 3.8 hold. Let $E_0 = E$ and $V_0 = V$. For each $i \in \mathbb{N}$, as long as there is a far edge in E_{i-1}, let f_i be the largest far edge in E_{i-1}, let F_i be the set of edges in E_{i-1} incident to $f_i \setminus e$, let $E_i = E_{i-1} \setminus F_i$, and let $V_i = V_{i-1} \setminus (f_i \setminus e)$. Suppose this process selects k far edges $f_1, \ldots, f_k$ and then there are no far edges left in $E_k = E \setminus (F_1 \cup \cdots \cup F_k)$.

For each $i \in [k]$, we have

$$\sum_{h \in F_i} d(h)x^2(h) \overset{\text{(a)}}{\geq} c' \max\{d(e), d(f_i)\} x^2(e) \overset{\text{(b)}}{\geq} \frac{c''}{n} \sum_{h \in F_i} d(h)x^2(e) \tag{3}$$

for some universal constants $c', c'' > 0$. Here (a) is by Lemma 3.6 and (b) is by Lemma 3.7. For the remaining edges E_k, which are all close, we have

$$\sum_{h \in E_k} d(h) x^2(h) \geq \frac{c'''}{n} \sum_{h \in E_k} d(h) x^2(e). \tag{4}$$

for some universal constant c''' by Lemma 3.8. Thus

$$\begin{aligned}
\sum_{h \in E} d(h) x^2(h) &= \sum_{i=1}^{k} \left(\sum_{h \in F_i} d(h) x^2(h) \right) + \sum_{h \in E_k} d(h) x^2(h) \\
&\overset{(c)}{\geq} \frac{c'' x^2(e)}{n} \sum_{i=1}^{k} \sum_{h \in F_i} d(h) + \frac{c''' x^2(e)}{n} \sum_{h \in E_k} d(h) \\
&\geq \frac{c D x^2(e)}{n},
\end{aligned}$$

for some universal constant $c > 0$, as desired. Here (c) applies inequalities (3) and (4). ■

3.2.1 Analysis of far edges

In this section we prove the two lemma's regarding far edges. We first prove Lemma 3.6 and restate the claim for the reader's convenience.

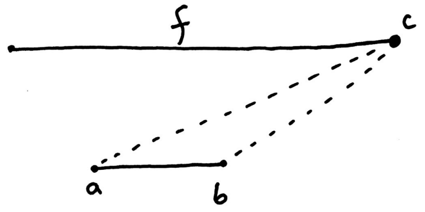
Lemma 3.6. *Let f be a far edge, and let F be the set of edges with both endpoints in $f \cup e$. Then*

$$\sum_{h \in F, h \neq e} d(h) x^2(h) \geq c \max\{d(e), d(f)\} x^2(e)$$

for some universal constant $c > 0$.

Proof. We have three cases.

(Case 1) f has length $d(f) > 2d(e)$

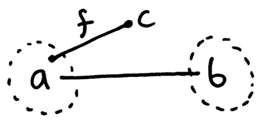


Then there is some endpoint $c \in f$ with distance at least $d(f)/4$ from both a and b. We have

$$\begin{aligned}
d(a,c)(x_a - x_c)^2 + d(b,c)(x_b - x_c)^2 &\overset{(a)}{\geq} \frac{d(f)}{4} \left((x_a - x_c)^2 + (x_b - x_c)^2 \right) \\
&\overset{(b)}{\geq} \frac{d(f)(x_a - x_b)^2}{4}
\end{aligned} \tag{5}$$

by (a) choice of c and (b) because the minimum over x_c is attained at the midpoint $x_c = (x_b + x_a)/2$.

(Case 2) f has length $d(f) \leq 2d(e)$, and some endpoint outside both B_a and B_b.

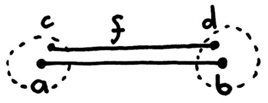


Let c be this endpoint. Then $d(a,c), d(c,b) \geq d(e)/4$, and similar to the derivation in line (5) above, we have

$$d(a,c)(x_a - x_c)^2 + d(b,c)(x_b - x_c)^2 \geq \frac{d(e)}{4}\Big((x_a - x_c)^2 + (x_b - x_c)^2\Big) \geq \frac{d(e)x^2(e)}{8}.$$

Moreover, $d(e) \geq d(f)/2$.

(Case 3) f has length $d(f) \leq 4d(e)$, one endpoint in B_a, and one endpoint in B_b.



f must be disjoint from e, since otherwise $f \setminus e$ would be contained in B_a or contained in B_b. Let c be the endpoint of f in B_a and let d be the endpoint of f in B_d. Then

$$\begin{aligned}
&d(a,d)(x_a - x_c)^2 + d(f)x^2(f) + d(b,c)(x_c - x_b)^2 \\
&\overset{(c)}{\geq} \frac{d(e)}{2}\Big((x_a - x_d)^2 + (x_c - x_d)^2 + (x_c - x_b)^2\Big) \\
&\overset{(d)}{\geq} \frac{d(e)}{6}(x_a - x_b)^2.
\end{aligned}$$

(c) is because each distance has length at least $d(e)/2$, because each edge being measured has one endpoint in B_a and the other in B_b. (d) minimizes over x_c and x_d. Moreover, $d(e) \geq d(f)/2$. ■

Next we prove Lemma 3.7, and first restate the claim for the reader's convience.

Lemma 3.7. *Let f be the largest edge such that $f \setminus e$ is not contained in B_a and not contained in B_b. Let h be any edge incident to f and not incident to e. Then*

$$d(h) \leq c \max\{d(f), d(e)\}.$$

for some universal constant $c > 0$.

Proof. If $h \setminus e$ is not contained in B_a and not contained in B_b, then we have $d(h) \leq d(f)$ by choice of h. Otherwise, if $h \setminus e$ is contained in B_a, then the endpoints of h are contained in $B_a \cup B_b$, so $d(h) \leq (3/2)d(e)$. ■

3.2.2 Analysis of close edges

For the final part of our analysis, we prove Lemma 3.8 regarding the contribution to the potential from close edges.

Lemma 3.8. *Suppose every edge is close. Then*

$$\sum_{f \in E} d(f)x^2(f) \geq c\frac{Dx^2(e)}{n}.$$

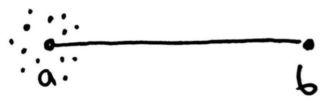
for some universal constant $c > 0$.

Proof. Let $U = V - e$ be the vertices besides the endpoints of e, and let $k = n - 2 = |U|$ be its cardinality. We first claim the following.

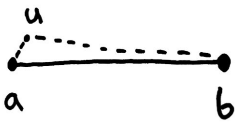
Claim 1. Either $U \subset B_a$ or $U \subset B_b$.

If not, and there is one such vertex c is closer to a, and another such vertex d is closer to b, then $\{c, d\}$ would be a far edge, a contradiction.

Without loss of generality suppose $U \subset B_a$.



We now claim that each point in U contributes energy proportional to the distance from u, in the following sense.



Claim 2. For all $u \in U$,

$$d(a,u)x^2(a,u) + d(b,u)x^2(b,u) \geq d(a,u)x^2(e).$$

We have

$$\begin{aligned} d(a,u)(x_u - x_a)^2 + d(b,u)(x_u - x_b)^2 &\overset{(a)}{\geq} 4d(a,u)\Big((x_u - x_a)^2 + (x_u - x_b)^2\Big) \\ &\overset{(b)}{\geq} d(a,u)(x_a - x_b)^2. \end{aligned} \tag{6}$$

Here (a) is because $d(b,u) \geq 3(a,b)/4 \geq 3d(a,u)$. (b) is because $(x_u - x_a)^2 + (x_u - x_b)^2$ is minimized at $x_u = (x_a + x_b)/2$.

Now we claim the following bound on the total energy contributed by edges between U and $\{a, b\}$. For a set of vertices $S \subseteq V$, let $D_S = \sum_{e \in E : e \subseteq S} d(e)$ denote the sum of distances over the metric restricted to S.

Claim 3. $\displaystyle\sum_{u \in U} \left(d(a,u)x^2(a,u) + d(b,u)x^2(b,u)\right) \geq \frac{1}{k} D_{U+a} x^2(e).$

To this end, we have

$$\begin{aligned} \sum_{u \in U} \left(d(a,u)x^2(a,u) + d(b,u)x^2(b,u)\right) &\overset{(c)}{\geq} \left(\sum_{u \in U} d(a,u)\right) x^2(e) \\ &\overset{(d)}{=} \frac{1}{k}\left(\sum_{u \in U} d(a,u)\right) x^2(e) + \frac{1}{k} D_U x^2(e) = \frac{1}{k} D_{U+a} x^2(e). \end{aligned}$$

Here (c) applies Claim 2 to each $u \in U$. (d) is because

$$D_U = \frac{1}{2} \sum_{u,v \in U} d(u,v) \overset{(e)}{\leq} \frac{1}{2} \sum_{u,v \in U} d(u,a) + d(a,v) = (k-1) \sum_{u \in U} d(u,a),$$

where (e) is by the triangle inequality.

The remaining distance to account for is between b and points in $U + a$.

Claim 4. $d(e)x^2(e) \geq \dfrac{4}{5(n-1)} \sum_{u \in U+a} d(b,u)x^2(e)$.

Indeed, for each $u \in U + a$, we have

$$\frac{5}{4}d(e) \overset{(f)}{\geq} d(e) + d(a,u) \overset{(g)}{\geq} d(b,u)$$

where (f) is because $u \in B_a$ and (g) is by the triangle inequality. Averaging over all $u \in U + a$ and multiplying gives Claim 4.

To conclude the proof, we have

$$\begin{aligned}\langle x^2, d\rangle &\overset{(h)}{\geq} \sum_{u \in U} d(a,u)x^2(a,u) + d(b,u)x^2(a,u) + d(e)x^2(e) \\ &\overset{(i)}{\geq} \frac{1}{k} D_{U+a}x^2(e) + \frac{4}{5(n-1)} \sum_{u \in U+a} d(b,u)x^2(e) \\ &\overset{(j)}{\geq} \frac{4}{5(n-1)} Dx^2(e).\end{aligned}$$

(h) drops the terms for edges between points in U. (i) is by Claim 3 and Claim 4. (j) recalls that $k = n - 2$ and observes that $D_{U+a} + \sum_{u \in U+a} d(b,u) = D$. ■

4 Spectral sparsification of kernels

In this section, we design and analyze spectral sparsifiers for geometric graphs induced by kernels.

4.1 Kernels and smoothness conditions

We study **kernel functions** $\kappa : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}_{\geq 0}$ that are nonnegative and symmetric ($\kappa(x,y) = \kappa(y,x)$). We develop algorithms for kernel functions that satisfy the following smoothness condition.

Condition 4.1. *For fixed parameters $\beta, t > 0$, and any three points x, y, z, we have*

$$\min\left\{\frac{1}{5}, \frac{\|x-y\|}{\|x-z\|}, \frac{\|x-y\|}{\|z-y\|}\right\} \leq \left(\beta \frac{\min\{\kappa(x,z), \kappa(y,z)\}}{\kappa(x,y)}\right)^{1/t}.$$

Remark 4.1. The choice of constant, 1/5, is primarily for ease of analysis. Other constants ≤ 1 would lead to similar results.

We only appeal to the technical conditions of Condition 4.1 in subsequent algorithms. For the remainder of this section, we list properties and simplifications of Condition 4.1 to broaden its applicability.

First, it is easy to see that the smoothness condition in Condition 4.1 is closed under natural operations like squaring a kernel taking the product of two kernels, with some change to the parameters (β, t).

Lemma 4.2. *Let $\kappa_1(x,y)$ be a kernel satisfying Condition 4.1 for parameters (β, t), and let κ_2 be a kernel defined by $\kappa_2(x,y) = \kappa_1^\alpha(x,y)$, for some $\alpha > 0$. Then κ_2 satisfies Condition 4.1 with parameters $(\beta, \alpha t)$.*

Lemma 4.3. *Let κ_1, κ_2 be two kernels where κ_1 satisfies Condition 4.1 for parameters (β_1, t_1) and $\kappa_2(x,y)$ satisfies Condition 4.1 for parameters (β_2, t_2). Then for any $\alpha \in (0,1)$, $\kappa(x,y) \overset{\text{def}}{=} \kappa_1(x,y) \cdot \kappa_2(x,y)$ satisfes Condition 4.1 for parameters $(\beta_1^\alpha \beta_2^{1-\alpha}, t_1^\alpha t_2^{1-\alpha})$.*

Condition 4.1 is somewhat technical as it is precisely the inequality directly invoked in the analysis. The following conditions are more intuitive and imply Condition 4.1.

Condition 4.2. *$\kappa(x, y)$ is nonincreasing in $\|x - y\|$, and for all x, y, z with $\|x - y\| \leq \|x - z\|$, we have*

$$\frac{\|x-y\|^t}{\|x-z\|^t} \leq \beta\left(\frac{\kappa(x,z)}{\kappa(x,y)}\right)^{1/t}. \tag{7}$$

Lemma 4.4. *Any kernel satisfying Condition 4.2 satisfies Condition 4.1 with the same parameters (β, t).*

Proof. For any three points x, y, z with (say) $\|x - z\| \leq \|y - z\|$.

$$\begin{aligned}
\min\left\{\frac{1}{5}, \frac{\|x-y\|}{\|x-z\|}, \frac{\|x-y\|}{\|y-z\|}\right\} &\leq \min\left\{\frac{\|x-y\|}{\|x-y\|}, \frac{\|x-y\|}{\|x-z\|}, \frac{\|x-y\|}{\|y-z\|}\right\} \\
&\overset{\text{(a)}}{\leq} \left(\beta\frac{\min\{\kappa(x,y),\kappa(x,z),\kappa(y,z)\}}{\kappa(x,y)}\right)^{1/t} \\
&\leq \left(\beta\frac{\min\{\kappa(x,z),\kappa(y,z)\}}{\kappa(x,y)}\right)^{1/t}
\end{aligned}$$

Here (a) invokes both monotonicity and inequality (7). ■

Condition 4.3. *$\kappa(x, y) = 1/f(\|x - y\|)$ for a real-valued function $f : \mathbb{R} \to \mathbb{R}$ with the following properties:*

(a) $f(x) > 0$ for all $x > 0$.

(b) f is nondecreasing.

(c) $\alpha^t f(x) \leq f(\alpha x)$ for all $x > 0$ and $\alpha \in (0, 1)$.

Lemma 4.5. *Suppose a kernel κ satisfies Condition 4.3 for a function f with parameter t. Then κ satisfies Condition 4.2 and (thereby) Condition 4.1 with parameters $(1, t)$.*

Proof. κ is nonincreasing by (b). Let x, y, z be three points with $\|x - y\| \leq \|x - z\|$. Let $q = \|x - y\|/\|x - z\| \leq 1$. We have

$$q^t = q^t\frac{f(\|x-z\|)}{f(\|x-z\|)} \overset{\text{(a)}}{\leq} \frac{f(q\|x-z\|)}{f(\|x-z\|)} = \frac{f(\|x-y\|)}{f(\|x-z\|)} = \frac{\kappa(x-z)}{\kappa(x-y)},$$

as desired. Here (a) is by property (c) of Condition 4.3. ■

Lemma 4.6. *Let $\kappa(x, y) = 1/f(\|x - y\|)$ for a polynomial f with nonnegative coefficients and maximum degree t. Then κ satisfies Condition 4.3 with parameter t.*

Proof. Clearly f is positive and nondecreasing. Write $f(x) = a_0 + a_1x + \cdots + a_tx^t$, where $a_0, a_1, \ldots, a_{t-1} \geq 0$ and $a_t > 0$. For any $\alpha \in (0, 1)$,

$$\begin{aligned}
\alpha^t f(x) &= a_0\alpha^t + a_1\alpha^t x + \cdots + a_t\alpha^t x^t \\
&\overset{\text{(a)}}{\leq} a_0 + a_1\alpha x + \cdots + a_t\alpha^t x^t = f(\alpha x),
\end{aligned}$$

where (a) is because $\alpha^t \leq \alpha^s$ for all $s \leq t$. ■

Via these simpler conditions, it is easy to see that some examples of smooth kernels satisfying Condition 4.1 are the following.

$$\kappa(x,y) = \frac{1}{\|x-y\|} \qquad (\beta,t) = (1,1) \tag{8}$$

$$\kappa(x,y) = \frac{1}{1+\|x-y\|^2} \qquad (\beta,t) = (1,2) \tag{9}$$

$$\kappa(x,y) = \frac{1}{\left(1+\|x-y\|^2\right)^{\alpha}} \qquad (\beta,t) = (1,2\alpha) \tag{10}$$

$$\kappa(x,y) = \frac{1}{1+\|x-y\|+\frac{1}{2}\|x-y\|^2} \qquad (\beta,t) = (1,2) \tag{11}$$

4.2 Kernel graphs and their effective resistance

Given a kernel $\kappa : V \times V \to \mathbb{R}_{\geq 0}$, the **kernel graph**, denoted G_κ, is defined to be the undirected graph on V where each edge $e = \{u,v\}$ has weight $\kappa(u,v)$. We let L_κ denote the Laplacian of G_κ. Note that G_κ and L_κ are dense, and takes $O(n^2)$ kernel evaluations to construct explicitly. The high level goal is to randomly construct a graph H that closely approximates G_κ using a nearly linear number of kernel valuations instead.

We want to apply Lemma 2.1 to sparsify kernel-weighted graphs. To apply Lemma 2.1, we need lower bounds on the effective resistance of each edge. Such a lower bound should be as large as possible to minimize the number of edges in the spectral sparsifier. At the same time, we want fairly simple lower bounds so that we can compute all $\approx n^2$ bounds implicitly in time proportional to n.

Consider the kernel graph $G_\kappa = (V, E, \kappa)$. For each edge $e = \{a,b\} \in E$, define

$$r(e) \overset{\text{def}}{=} \kappa(e) + \frac{1}{2}\sum_{c \notin e} \min\{\kappa(a,c), \kappa(c,b)\}.$$

Lemma 4.7. *Each edge $e \in E$ has effective resistance (eff. resist. e) $\leq \frac{1}{r(e)}$.*

Proof. Let $e = \{a,b\}$, and let $x \in \mathbb{R}^V$ with $x_a = 0$ and $x_b = 1$. It suffices to show that

$$\sum_{\{c,d\}\in E} \kappa(c,d)(x_c - x_d)^2 \geq r(e). \tag{12}$$

For each vertex $c \neq e$, we have

$$\begin{aligned}&(x_a - x_c)^2\kappa(a,c) + (x_b - x_c)^2\kappa(b,c)\\ &\geq \left((x_a - x_c)^2 + (x_b - x_c)^2\right)\min\{\kappa(a,c), \kappa(b,c)\},\end{aligned}$$

and

$$(x_a - x_c)^2 + (x_b - x_c)^2 = x_c^2 + (1 - x_c)^2 \geq 1/2.$$

Summing over all such c, and then including the term for e, gives the desired inequality (12). ■

Remark 4.8. Obviously, Lemma 4.7 holds for any non-negative weighted graph as well.

4.3 Gaussian random variables and projections

We use the Gaussian distribution to define random projections to lower-dimensional spaces. Recall that a Gaussian random variable $x \in \mathbb{R}$ with mean μ and variance σ^2 (denoted $x \sim \mathcal{N}(\mu, \sigma^2)$) is defined by the probability density function

$$x \mapsto \frac{1}{\sqrt{2\pi\sigma^2}} e^{-(x-\mu)^2/2\sigma^2}.$$

We abbreviate the standard normal distribution $\mathcal{N} \overset{\text{def}}{=} \mathcal{N}(0,1)$. For $d \in \mathbb{N}$, we let $\mathcal{N}^d$ denote the d-variate standard normal distribution. That is, $x \sim \mathcal{N}^d$ denotes a random vector $x \in \mathbb{R}^d$ where each coordinate x_i is independently distributed as $\mathcal{N}^d$.

We employ the following well-known facts about Gaussian random variables.

Lemma 4.9. *Let $x \sim \mathcal{N}$ be a Gaussian random variable, and let $t > 0$. Then:*

$$\mathrm{P}[x \geq t] = \mathrm{P}[x \leq -t] \leq \frac{1}{t\sqrt{2\pi}} e^{-t^2/2}. \tag{13}$$

$$\mathrm{P}[0 \leq x \leq t] = \mathrm{P}[-t \leq x \leq 0] \leq \frac{t}{\sqrt{2\pi}}. \tag{14}$$

Lemma 4.10. *Let $x \in \mathbb{R}^d$, and let $a \sim \mathcal{N}^d$ be a random Gaussian vector. Then $\langle a, x\rangle$ is distributed as a Gaussian random variable with mean 0 and standard deviation $\|x\|^2$; that is, $\langle a, x\rangle \sim \mathcal{N}\left(0, \|x\|^2\right)$.*

We frequently apply Lemma 4.9 to the Gaussian projections in Lemma 4.10.

4.4 1-smooth kernels and line embeddings

In this section, we present and analyze the spectral sparsifier for kernel graphs G_κ where the kernel satisfies Condition 4.1 with parameters $(\beta, 1)$ where $\beta \geq 1$. The results in this section are to some extent subsumed by subsequent results for general t, but we highlight two differences. First, the algorithm here is (in our opinion) much simpler, and a good setting to introduce many of the new ideas, because it does not involve any data structures. Second, and somewhat more technical, is that the algorithm in this section has no dependence on the geometric spread of the point set.

4.4.1 Algorithm

The algorithm is called `randomly-ranked-spectral-sample(`$V, \kappa, \beta, \epsilon$`)` (abbreviated `RRSS`) and pseudocode is given in Figure 3. The algorithm was described at a high level in Section 1.3.2, and now we describe it more precisely.

Given a set of points V in $\mathbb{R}^d$, we repeat the following randomized process for $i = 1, \ldots, k$ where $k = O(\log n)$. We draw a random Gaussian vector $a_i \sim \mathcal{N}^d$, and compute the projections $\langle a_i, v\rangle$ for all $v \in V$. We order the points in v in increasing order of $\langle a_i, v\rangle$, and let $\mathrm{rank}_i(v)$ denote their index in this order. We then apply Lemma 2.1. We sample $O\left(n \log^2(n)/\epsilon^2\right)$ edges with replacement in proportion to $\sum_{i=1}^k 1/|\mathrm{rank}_i(x) - \mathrm{rank}_i(y)|$ for each edge $e = \{x, y\}$. We scale up the weights of sampled edges as described in Lemma 2.1.

```
randomly-ranked-spectral-sample($V$,$\kappa$,$\beta$,$\epsilon$)

/* Abbreviated RRSS($V$,$\kappa$,$\beta$,$\epsilon$). Let $n = \log|V|$                                */

1. for $i = 1, \ldots, k = O(\log n)$
    A. draw $a_i \sim \mathcal{N}^d$
    B. order $V$ in increasing order of $\langle a_i, v\rangle$
    C. let $\mathrm{rank}_i(v)$ = index of $v$ in this order
2. apply Lemma 2.1 and importance sample $O\left(n\log^2(n)/\epsilon^2\right)$ edges with repetition in proportion to

       $p_e = \frac{1}{\log n}\sum_{i=1}^{k}\frac{1}{|\mathrm{rank}_i(a) - \mathrm{rank}_i(b)|}$ for each edge $e = \{a,b\}$
```

Figure 3: A randomized nearly linear time algorithm for sparsifying a smooth kernel weighted graph.

4.4.2 Analysis

We now analyze RRSS. The following key lemma is the primary motivation for the algorithm. It states that with constant probability, the difference in rank in the endpoints of an edge is an underestimate for $1/(\text{eff. resist. } e)\kappa(e)$.

Lemma 4.11. *Let $e = \{x, y\}$ be an edge. Then*

$$|\mathrm{rank}_i(x) - \mathrm{rank}_i(y)| \le \frac{c_1\beta}{(\textit{eff. resist. } e)\kappa(e)} \textit{ with probability } c_2,$$

where $c_1, c_2 > 0$ are universal constants.

Proof. Call a vertex z **close** if $\max\left\{\langle a, x - z\rangle^2, \langle a, y - z\rangle^2\right\} \le \|x - y\|^2$.

Claim 1. For each $z \notin e$,

$$\mathrm{P}[z \textit{ close}] \le 2\beta\frac{\min\{\kappa(x,z), \kappa(x,y)\}}{\kappa(x,y)}.$$

Without loss of generality, suppose z is further from x than y (i.e., $\|x - z\| > \|y - z\|$). Then

$$\begin{aligned}\mathrm{P}[z \text{ close}] &= \mathrm{P}\left[\langle a, x - z\rangle^2 \le \|x - y\|^2\right] \\ &\overset{(a)}{\le} 2\min\left\{\frac{1}{2}, \frac{\|x - y\|}{\|x - z\|}\right\} \overset{(b)}{\le} 2\beta\frac{\min\{\kappa(x,z), \kappa(z,y)\}}{\kappa(x,y)}.\end{aligned}$$

Here (a) is because $\langle a, x - z\rangle$ is distributed as a Gaussian with mean 0 and standard deviation $\|x - z\|$. (b) is because κ satisfies Condition 4.1 with $t = 1$.

Claim 2. For any $t > 1$, we have

$$\mathrm{P}\left[(\textit{\# close vertices not in } e) > \frac{4t\beta(r(e) - 1)}{\kappa(x,y)}\right] \le \frac{1}{t}.$$

Claim 2 follows from applying Markov's inequality to the number of close vertices not in e. Here we observe that

$$\mathrm{E}[\text{\# close vertices not in } e] \overset{\text{(c)}}{\le} \sum_{z\notin e} 2\beta \frac{\min\{\kappa(x,z),\kappa(z,y)\}}{\kappa(x,y)} \overset{\text{(d)}}{=} \frac{4\beta(r(e)-\kappa(e))}{\kappa(x,y)}.$$

by (c) Claim 1 and (d) definition of $r(e)$.

Claim 3. With probability $> .68$,

$$|\mathrm{rank}_i(x) - \mathrm{rank}_i(y)| \le 1 + (\text{\# } z \text{ close}, z \notin e).$$

Observe that if $\langle a, x-y\rangle^2 < \|x-y\|^2$, then a vertex z has rank between x and y only if z is close. $\langle a, x-y\rangle$ is distributed as a centered Gaussian with standard devation $\|x-z\|$, and a Gaussian deviates by less than its standard deviation with probability $> .68$.

With (say) probability $> 1/2$, then, both the events in Claim 2 (for $t = 1/.18$) and Claim 3 hold. Then

$$|\mathrm{rank}_i(x) - \mathrm{rank}_i(y)| \overset{\text{(e)}}{\le} 1 + 2(\text{\# } z \text{ close}, z \notin e) \overset{\text{(f)}}{\le} O\left(\frac{\beta r(e)}{\kappa(e)}\right) \overset{\text{(g)}}{\le} O\left(\frac{\beta}{(\text{eff. resist. } e)\kappa(e)}\right)$$

by (e) Claim 2, (f) Claim 3, and (g) Lemma 4.7, as desired. ∎

Theorem 4.12. *Let $\kappa : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}_{\ge 0}$ be a symmetric function satisfying Condition 4.1 with $(\beta, 1)$ with $\beta \ge 1$. Let Q denote the time to evaluate κ. Let $V \subset \mathbb{R}^d$ be a set of n points. Then with high probability,* `randomly-ranked-spectral-sample(`V,κ,β,ϵ`)` *returns a $(1\pm\epsilon)$-spectral approximation of G_κ with $O\left(\beta n \log^2(n)(\log(n)+Q)/\epsilon^2\right)$ time, one can compute a $(1\pm\epsilon)$-spectral approximation of G_κ with $O\left(\beta n \log^2(n)/\epsilon^2\right)$ total edges and maximum degree $O\left(\beta \log^2(n)/\epsilon^2\right)$.*

Proof. We analyzed the algorithm `ranked-importance-sample`, given in Figure 3. For each edge $e = \{x,y\} \in E$, let

$$p_e = \frac{\beta}{\log n} \sum_{i=1}^{k} \frac{1}{|\mathrm{rank}_i(x) - \mathrm{rank}_i(y)|}.$$

Claim 1. With high probability, $p_e \ge \kappa(e)(\text{eff. resist. } e)$.

For each $i \in [k]$, we have

$$\frac{1}{|\mathrm{rank}_i(x) - \mathrm{rank}_i(y)|} \ge \frac{\kappa(e)(\text{eff. resist. } e)}{c_1\beta} \text{ with probability } c_2$$

by Lemma 4.11, for universal constants $c_1, c_2 \in (0,1)$. In particular, over $2\log(n)/c_1c_2$ samples, the above holds for at least $2\log(n)/c_1$ indices i in expectation. As the ranks rank_i are independent across i, the above holds for $\log(n)/c_1$ indices i with high probability, in which case we have

$$p_e = \frac{\beta}{\log n} \sum_i \frac{1}{|\mathrm{rank}_i(x) - \mathrm{rank}_i(y)|} \ge \kappa(e)(\text{eff. resist. } e),$$

as desired.

Claim 2. $\displaystyle\sum_{e\in E} p_e \le O(\beta n \log n)$.

For each i, we have

$$\begin{aligned}\sum_{\{x,y\}\in E} \frac{1}{|\mathrm{rank}_i(x) - \mathrm{rank}_i(y)|} &= \sum_{x\in V} \sum_{\substack{y\in V \text{ where} \\ \mathrm{rank}_i(y)>\mathrm{rank}_i(x)}} \frac{1}{\mathrm{rank}_i(y) - \mathrm{rank}_i(x)} \\ &= \sum_{i=1}^{n} \sum_{j=i+1}^{n} \frac{1}{j-i} \leq n\log(1+n). \end{aligned} \tag{15}$$

Thus

$$\sum_{e\in E} p_e \overset{\text{(a)}}{=} \frac{\beta}{\log n} \sum_i \sum_{\{x,y\}\in E} \frac{1}{|\mathrm{rank}_i(x) - \mathrm{rank}_i(y)|} \leq O(\beta n \log n)$$

by (a) interchanging sums and (b) substituting the inequality obtained in (15), as desired.

By Lemma 2.1, we importance sample $O\left(\beta n \log^2(n)/\epsilon^2\right)$ edges with replacement in proportion to $\{p_e : e \in E\}$ and produce a $(1 \pm \epsilon)$-spectral approximation with high probability. It is easy to sample in proportion to $\{p_e : e \in E\}$ in $O(\log n)$ time per edge because of the particularly simple and symmetric structure of the p_e's[4]. Each sampled edge requires one kernel evaluation. ■

4.5 Generally smooth kernels

In this section, we describe and analyze an algorithm for kernels satisfying Condition 4.1 with parameter t greater than 1.

4.5.1 Algorithm

A high-level overview of the algorithm was given in Section 1.3.2. Here we explain the algorithm in detail. The algorithm is called `randomly-projected-spectral-sample`(V,κ,t,β), abbreviated `RPSS`(V,κ,t,β), and pseudocode is given in Figure 4.

The sparsification algorithm applies importance sampling (with replacement) to produce a spectral sparsifier of the kernel graph, applying in particular Lemma 2.1. Most of the algorithmic effort is in (implicitly) generating estimates of the spectral importance of each edge, in time proportional to the number of vertices rather than the number of edges.

For a kernel satisfying Condition 4.1 with parameter t, we repeat the following randomized process $k = O\left(e^{O(t)}\log(n)\right)$ times. For $i \in [k]$, let $\pi_i : \mathbb{R}^d \to \mathbb{R}^t$ be a random projection, where for each output dimension $j \in [t]$, we have $(\pi_i(x))_j = \langle a_j, x\rangle$ for a randomly sampled Gaussian vector $a_j \sim \mathcal{N}^d$. We then build a quadtree over the projected point set $\{\pi_i(v), v \in V\}$.

Quadtrees are a simple geometric data structure well-suited for low-dimensional point sets. Here we give a brief overview and refer the reader to [11, 25] for a more detailed description. A quadtree on $\mathbb{R}^t$ is a hierarchical family of grids on $\mathbb{R}^t$. For each integer ℓ (of interest), it overlays a grid of cells with side length 2^ℓ. We call this the **grid at level ℓ**. The grids across levels ℓ are lined up so that each cell with side length 2^ℓ splits into 2^t cells of length $2^{\ell-1}$ in the next smaller grid at level $\ell - 1$. For each cell, we record the number of points in that cell. The smallest level ℓ included in the quadtree is the largest ℓ such that every point is in its own cell. The largest level ℓ is chosen to be the smallest ℓ such that every point is contained in the same cell. Translating the grids appropriately, this can be accomplished so that there are $O(\log \Phi_\infty)$ levels between the smallest and largest levels, where Φ_∞ is the L_∞-spread of the point set.

[4]In particular, we generate $i \in [k]$ uniformly at random, and two distinct indices $j_1, j_2 \in [n]$ in proportion to $1/|k_1 - k_2|$, in which case we return the vertices with rank k_1 and k_2 in the ith embedding.

```
randomly-projected-spectral-sample($V,\kappa,t,\beta$)

// Abbreviated RPSS($V,\kappa,t,\beta$)

1. for $i = 1,\dots,k = O\left(e^{O(t)}\log(n)\right)$
    A. define $\pi_i : \mathbb{R}^d \to \mathbb{R}^t$ by $(\pi_i(x))_j = \langle a_j, x\rangle$ for $a_j \sim \mathcal{N}^d$
    B. build a quadtree over $\{\pi_i(v) : v \in V\}$ per Lemma 4.13 with $L_i$ levels.

/* randomly construct a sparse weighted (multi-)graph where edges are importance sampled based on the
   quadtrees                                                                                     */

2. let $H = (V,\emptyset)$ be an empty graph over $V$, and let $\alpha = O\left(\log(n)/\epsilon^2\right)$

3. repeat $m = \alpha n \sum_{i=1}^{k} L_i$ times
    A. sample $i \in [k]$ and $x \in X$ uniformly at random
    B. sample a level $\ell$ in the $i$th quadtree uniformly at random
    C. sample a vertex $y \in A_i(x,\ell)$ uniformly at random                 // cf. Definition 4.14
    D. compute $p(x,y) = \sum_{i=1}^{k} \sum_{\substack{\ell: y\in A_i(x,\ell) \\ \text{and } x\in A_i(y,\ell)}} \frac{1}{|A_i(x,\ell)|} + \frac{1}{|A_i(y,\ell)|}$
    E. add edge $\{x,y\}$ to $H$ with weight $\kappa(x,y)/\alpha p(x,y)$

4. return $H$
```

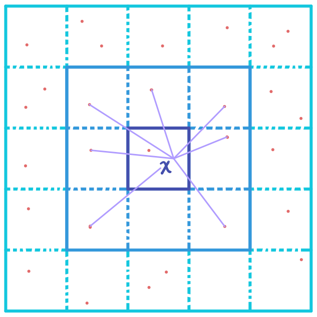
Figure 4: Sparsifying a kernel graph for kernels in $\mathbb{R}^d$ satisfying Condition 4.1 in $\tilde{O}(nd)$ time.

Lemma 4.13. *Let $X \subseteq \mathbb{R}^t$ be a set of n points with L_∞-spread $\Phi_\infty \overset{\text{def}}{=} \frac{\max_{x \neq y} \|x-y\|_\infty}{\min_{x\neq y}\|x-y\|_\infty}$. The one can build a shifted quadtree over X with $O(\log(\Phi_\infty))$ levels in $O(nt \log n + n \log \Phi_\infty)$ time.*

Once the projected points are placed in a quadtree, we describe their relative placement with the following terminology.

Definition 4.14. *For a fixed quadtree over a projected point set $\{\pi_i(v), v \in V\}$, we say that two cells are* **adjacent** *if they are at the same level and are touching (or the same). We say that two vertices are* **adjacent** *at a level ℓ if they are in adjacent cells at level ℓ. We define $\boldsymbol{A_i(x, \ell)}$ as the set of vertices adjacent to x at level ℓ in the ith quadtree.*

Let L be the total number of levels across all quadtree, and let $\alpha = O(1/\epsilon^2)$. We build a random graph H by sampling $m = \alpha n L$ edges with replacement based on the quadtree. For each sample, we first select a vertex $x \in V$ and a trial $i \in [k]$ uniformly at random. Then we select a level ℓ in the ith quadtree uniformly at random. Then we select a vertex $y \in A_i(x, \ell)$ uniformly at random. Clearly it is easy to sample i and x in $O(\log n)$ time. With $O\left(e^{O(t)}\right)$ overhead in the preprocessing step, we can augment the quadtree with enough information to sample $y \in A_i(x, \ell)$ in $O(\log n)$ time as well.



We add the edge $\{x, y\}$ to the graph with its weight scaled up as follows. First, observe that $\{x, y\}$ is selected in proportion to the quantity

$$p(x, y) = \sum_{A_i(x,\ell) \ni y} \frac{1}{|A_i(x, \ell)|} + \sum_{A_i(y,\ell) \ni x} \frac{1}{|A_i(y, \ell)|},$$

which can be computed quickly by, for each $i \in [k]$, doing a binary search for the first level ℓ where $x \in A_i(y, \ell)$ and $y \in A_i(x, \ell)$. We add the edge $\{x, y\}$ with weight $\kappa(x, y)/\alpha p(x, y)$ to H.

The sampling process we just described is a concrete implementation of the sparsification procedure described in Lemma 2.1. If $p(e)$ is an upper bound on (eff. resist. e)$\kappa(e)$ for all edges e, then H will be a $(1 \pm \epsilon)$-spectral sparsifier of G_κ with high probability.

4.5.2 Analysis

Definition 4.15. *For an edge e, the* **ideal level** *of e, denoted $\ell(e)$, is defined as*

$$\ell(e) = \lceil \log_2 \|x - y\| \rceil.$$

Lemma 4.16. *For each $i \in [k]$, with probability $\geq e^{-O(t)}$,*

$$|A_i(x, \ell(e))| + |A_i(y, \ell(e))| \leq \frac{\beta e^{O(t)}}{\kappa(e)(\textit{eff. resist. } e)}$$

Proof. For simplicity, we say that points or cells are "adjacent" if in particular they are adjacent at ideal level $\ell(e)$. We call a point z **close** if both $\|\pi(x)-\pi(z)\|_\infty$ and $\|\pi(y)-\pi(z)\|_\infty$ are $\le 5\|x-y\|$.

Claim 1. For any point z, we have

$$\mathrm{P}[z \textit{ is close}] \le 4^t \beta \frac{\min\{\kappa(x,z),\kappa(y,z)\}}{\kappa(x,y)}.$$

Without loss of generality suppose z is further from x (i.e., $\|x-z\| > \|y-z\|$). Then

$$\begin{aligned}
\mathrm{P}[\|\pi(x)-\pi(z)\|_\infty \le 5\|x-y\|] &\overset{(a)}{=} \prod_{i=1}^{k} \mathrm{P}\left[\langle a_i, x-z\rangle^2 \le 5\|x-y\|^2\right] \\
&\overset{(b)}{\le} \min\left\{1, \left(\frac{5\|x-y\|}{\|x-z\|}\right)\right\}^t \\
&= 5^t \min\left\{\frac{1}{5}, \frac{\|x-y\|}{\|x-z\|}, \frac{\|x-y\|}{\|z-y\|}\right\}^t \\
&\overset{(c)}{\le} 5^t \beta \frac{\min\{\kappa(x,z),\kappa(y,z)\}}{\kappa(x,y)}.
\end{aligned}$$

Here (a) is by independence of each embedding. (b) is because each $\langle a_i, x-z\rangle$ is distributed as a centered Gaussian with standard deviation $\|x-y\|$. (c) is by assumption on κ.

Claim 2. With probability of error $< 4^{-t}$, we have (# close vertices) $\le \dfrac{20^t \beta r(e)}{\kappa(e)}$

We have

$$\begin{aligned}
\mathrm{P}\left[(\text{\# close vertices}) \ge \frac{20^t\beta r(e)}{\kappa(e)}\right] &\overset{(d)}{\le} \frac{\mathrm{E}[\text{\# close vertices}]\kappa(e)}{20^t \beta r(e)} \\
&\overset{(e)}{\le} \frac{5^t \sum_z \min\{\kappa(x,z),\kappa(y,z)\}}{20^t \kappa(x,y) r(e)} \overset{(f)}{\le} \frac{1}{4^t}
\end{aligned}$$

by (d) Markov's inequality, (e) linearity of expectation and Claim 1, and (f) definition of $r(e)$.

Claim 3. With probability $\ge 2^{-t}$, we have $\|\pi(x)-\pi(y)\|_\infty \le \|x-y\|$.

We have

$$\begin{aligned}
\mathrm{P}[\|\pi(x)-\pi(y)\|_\infty \le \|x-y\|] &\overset{(g)}{=} \prod_{i=1}^{t} \mathrm{P}[|\langle a_i, x-y\rangle| \le \|x-y\|] \\
&\overset{(h)}{\ge} \left(1-\sqrt{2/e\pi}\right)^t > 2^{-t}.
\end{aligned}$$

Here (g) is because each $a_i \sim \mathcal{N}^d$ is drawn independently. (h) applies Lemma 4.9 to $\langle a_i, x-y\rangle$, which is distributed as a centered Gaussian with standard deviation $\|x-y\|$ (Lemma 4.10).

Claim 4. If $\|\pi(x)-\pi(y)\|_\infty < \|x-y\|$, and a point z is adjacent to x and y, then z is close.

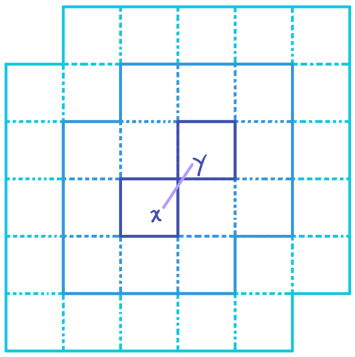
Indeed, suppose $\|\pi(x)-\pi(y)\|_\infty < \|x-y\|$, and let z be adjacent to x. Then

$$\|\pi_i(x)-\pi_i(z)\|_\infty \le 2\cdot 2^{\ell(e)} \le 4\|x-y\|,$$

and

$$\|\pi_i(y)-\pi_i(z)\|_\infty \le \|\pi_i(x)-\pi_i(y)\|_\infty + \|\pi_i(x)-\pi_i(y)\|_\infty \le 5\|x-y\|.$$

In particular, z is close. Symmetrically, if z is instead adjacent to y, z is again close.



To complete the proof, we have that with probability of success $\geq e^{-O(t)}$, both Claim 2 and Claim 3 hold. Then

$$\begin{pmatrix}\text{\# points adjacent}\\ \text{to } x \text{ or } y\end{pmatrix} \overset{\text{(i)}}{\leq} (\text{\# close vertices}) \overset{\text{(j)}}{\leq} \frac{\beta e^{O(t)} r(e)}{\kappa(e)} \overset{\text{(k)}}{\leq} \frac{\beta e^{O(t)}}{\kappa(e)(\text{eff. resist. } e)},$$

as desired. Here (i) is by Claim 4, (j) is by Claim 2, and (k) is by Lemma 4.7. ■

Theorem 4.17. *With high probability,* RPSS *returns a weighted graph H that is a $(1 \pm \epsilon)$-spectral sparsifier of G_κ and has $O\left(n\beta e^{O(t)} \log(n) \log(\Phi)/\epsilon^2\right)$ edges in $O\left(n\left(\log(\Phi) + \log(n)\right)\left(\log(n) + \log\log(\Phi) + Q\right)e^{O(t)}/\epsilon^2\right)$ randomized time.*

Proof. For each $e \in E$, and each trial $i \in [k]$, define

$$p_e = \frac{\beta e^{O(t)}}{\log n} \sum_i \sum_{\text{levels } \ell} \left(\frac{1}{|A_i(x,\ell)|} + \frac{1}{|A_i(y,\ell)|} \text{ if } e \subseteq A_i(x,\ell) \cap A_i(y,\ell) \right).$$

Claim 1. For all $e \in E$, with high probability, $p(e) \geq$ (eff. resist. $e)\kappa(e)$.

Fix $e = \{x, y\} \in E$. By Lemma 4.16, for each trial i, we have

$$y \subseteq A_i(x, \ell(e)),\, x \in A_i(y, \ell(e)), \text{ and } \frac{1}{|A_i(x,\ell)|} + \frac{1}{|A_i(y,\ell)|} \geq \frac{e^{-O(t)}}{\beta} \kappa(e)(\text{eff. resist. } e) \qquad (16)$$

with probability $\geq e^{-O(t)}$. Over $k = O\left(e^{O(t)} \log(n)\right)$ independent trials, the event (16) occurs at least $2\log(n)$ times in expectation, and at least $\log(n)$ times with high probability. When the latter occurs, we have $p_e \geq \kappa(e)(\text{eff. resist. } e)$.

Claim 2. $\displaystyle\sum_{e \in E} p_e \leq \frac{\beta e^{O(t)} n}{\log n} \begin{pmatrix}\textit{total \# levels over}\\ \textit{all quadtrees}\end{pmatrix}.$

For each i, we have

$$\begin{aligned}&\sum_{\{x,y\} \in E} \sum_\ell \left(\frac{1}{|A_i(x,\ell)|} + \frac{1}{|A_i(y,\ell)|} \text{ if } \{x,y\} \subseteq A_i(x,\ell) \cap A_i(y,\ell) \right) \\ &\overset{\text{(a)}}{=} \sum_x \sum_\ell \sum_{y \in A_i(x,\ell)} \frac{1}{|A_i(x,\ell)|} = \sum_x \sum_\ell 1 = n \begin{pmatrix}\text{\# levels in the}\\ i\text{th quadtree}\end{pmatrix}.\end{aligned}$$

where (a) interchanges sums. Summing up the above equality over all i and then multiplying both sides by $\beta e^{O(t)}/\log n$ gives the desired inequality.

Claim 3. With high probability, for all i, we have

$$\log\left(\frac{\max_{x,y}\|\pi_i(x)-\pi_i(y)\|_\infty}{\min_{x,y}\|\pi_i(x)-\pi_i(y)\|_\infty}\right) \le \log(\Phi) + O\left(\frac{\log n}{t}\right).$$

To prove Claim 3, fix i. Let $\delta_0 = \min_{x\neq y}\|x-y\|$ be the minimum distance between any two points, and let $\delta_1 = \max_{x\neq y}\|x-y\|$ be the maximum distance. For any two points x and y, each coordinate of $\pi_i(x)-\pi_i(y)$ is distributed as a Gaussian with mean 0 and standard deviation $\|x-y\|$. For any $\alpha > 1$, we have

$$\mathrm{P}[\|\pi_i(x)-\pi_i(y)\|_\infty \ge \alpha\delta_1] \le \frac{\sqrt{2}}{\alpha\sqrt{\pi}}e^{-\alpha^2/2}.$$

In particular, for $\alpha = O\left(\sqrt{t\log n}\right)$, we have

$$\mathrm{P}\left[\|\pi_i(x)-\pi_i(y)\|_\infty \ge O\left(\sqrt{t+\log n}\right)\delta_1\right] \le e^{-O\left(t+\log n\right)}.$$

On the other hand, for any $\alpha < 1$, we have

$$\mathrm{P}[\|\pi_i(x)-\pi_i(y)\|_\infty \le \alpha\delta_0] \le \left(\alpha\sqrt{\frac{2}{\pi}}\right)^t.$$

For $\alpha = n^{-O(1/t)}$, we have

$$\mathrm{P}\left[\|\pi_i(x)-\pi_i(y)\|_\infty \le \delta_0/n^{O(1/t)}\right] \le e^{-O\left(t+\log n\right)}.$$

Taking a union bound over all n^2 pairs, we have

$$\mathrm{P}\left[\frac{\max_{x,y}\|\pi_i(x)-\pi_i(y)\|_\infty}{\min_{x,y}\|\pi_i(x)-\pi_i(y)\|_\infty} \ge O\left(n^{O(1/t)}\sqrt{t+\log n}\right)\Phi\right] \le e^{-O\left(t+\log n\right)}.$$

Taking the union bound over all i, we have that the above holds for all i with probability of error $\le e^{-O\left(t+\log n\right)}$. This proves Claim 3.

With high probability, both Claim 1 (for all $e \in E$) and Claim 3 hold. Then each quadtree has height $O\left(\log(\Phi)+\log(n)/t\right)$, and the total number of levels across all quadtrees is at most $O\left(e^{O(t)}\log(n)\left(\log(\Phi)+\log(n)\right)\right)$. By Claim 2 we have

$$\sum_{e\in E} p(e) \le \beta e^{O(t)} n\left(\log(\Phi)+\log(n)\right).$$

The desired graph H now follows from Lemma 2.1.

As per the running time, in the high probability event of success, each quadtree takes $O\left(ndt + n\left(\log(\Phi)+\log(n)\right)e^{O(t)}\right)$ time to build, and there are $k = O\left(e^{O(t)}\log(n)\right)$ many quadtrees. Thus all the quadtrees take $O\left(n\left(d+\log(\Phi)+\log(n)\right)e^{O(t)}\log(n)\right)$ time to build. We sample $O\left(\log(n)\sum_e p_e/\epsilon^2\right) = O\left(n\left(\log(\Phi)+\log(n)\right)\beta e^{O(t)}/\epsilon^2\right)$ edges, and each edge sample takes $O\left(t+\log(n)+\log\log(\Phi)+Q\right)$ time between sampling the edge and evaluating the kernel. Thus the overall running time is dominated by the sampling step, and is

$$O\left(n\left(\log(\Phi)+\log(n)\right)\left(\log(n)+\log\log(\Phi)+Q\right)e^{O(t)}/\epsilon^2\right),$$

as desired. ■

4.6 Density queries

In this section, we consider the problem of kernel density estimation. The density estimation problem is discussed extensively in Section 1.2 and an important primitive for many kernel-based computational problems. The problem is posed as a data structure problem and we recall the setup. Let $V \subseteq \mathbb{R}^d$ be a fixed point set. We want to process V to efficiently serve queries of the following form. Given a point $q \in \mathbb{R}^d$, compute the sum

$$\sum_{v \in V} \kappa(q, v). \tag{17}$$

Following previous work, we seek a $(1 \pm \epsilon)$-relative approximation to the above with high probability.

[4] gave the following bounds for an essentially similar notion of smoothness. [4] showed that with

$$O\Big(n\big(d + \log(n) + \log(\Phi)\big) \log(n) e^{O(t)} / \epsilon^2\Big)$$

space and preprocessing time, one can construct a data structure that computes $(1 \pm \epsilon)$-multiplicative approximations to (17) in

$$\beta e^{O(t)} \log(n)\big(\log(n) + \log(\Phi) + d\big)/\epsilon^2$$

time. The significance of these bounds is discussed in Section 1.2.

We describe an alternative data structure for the kernel density estimation problem which falls out naturally from the spectral sparsifier construction. The high level ideas are discussed Section 1.3.2, which we briefly review. The sum in (17) can be interpreted as the weighted degree of the kernel graph of $V + q$. Since the weighted degree is the singleton cut of $\{q\}$, it is preserved by the spectral sparsifiers constructed in this work. Thus one algorithm to approximate (17) is to build the spectral sparsifier of Section 4.5 over $V + q$, and then return the weighted degree of q. Of course, an entire spectral sparsifier, and in particular, the effort to sparsify the edges between V, is unnecessary.

Theorem 4.18. *With $O\Big(n\big(d + \log(n) + \log(\Phi)\big) \log(n) e^{O(t)}\Big)$ space and preprocessing time, one can build a data structure that computes $(1 \pm \epsilon)$-approximate to (β, t)-smooth kernel density queries with high probability (for each query) in*

$$O\Big(\beta e^{O(t)} \big(\log(n) + \log(\Phi)\big)\big(\log(n) + \log\log(\Phi) + Q\big)/\epsilon^2\Big)$$

time.

Proof. We build a family quadtrees over random projections of V onto $\mathbb{R}^t$ as in Section 4.5. Upon a query q, we add the cells corresponding to the projections of q to the many quadtree. We then sample edges incident to q as in the construction of Section 4.5. Repeatedly, we uniformly sample a quadtree and a cell containing q. We select a point v uniformly at random from those adjacent to q. We then include $\kappa(x, y)$, scaled in inverse proportion to the total number of vertices adjacent to q at that level in that quadtree. The analysis of this data structure is essentially the same as the analysis in Section 4.5, and therefore omitted. Here one can bypass the matrix concentration inequalities and instead apply standard Chernoff bounds, as we are only concerned with estimating a single sum.

As described above, we require $O\Big(n\big(d + \log(\Phi) + \log(n)\big) e^{O(t)} \log(n)\Big)$ to construct the quadtrees over random projections of V. The space, beyond the original input points, is $O\Big(n\big(d + \log(\Phi) + \log(n)\big) e^{O(t)} \log(n)\Big)$ on account of the quadtrees. We also need $O\Big(\big(\log(\Phi)+\log(n)\big)\beta e^{O(t)}/\epsilon^2\Big)$ samples for each query q, and each sample takes $O\big(t+\log(n)+\log\log(\Phi)+Q\big)$ between sampling the point and then evaluating the kernel. ■

For error probability $\leq 1/\mathrm{poly}(n)$, the preprocessing and space bounds of Theorem 4.18 improve that of [4] by a $O\big(1/\epsilon^2\big)$-multiplicative factor. The query time bounds are fairly similar although the dependence on Q here is less by a $\log(n)$ factor. (A factor of $\log(n)Q$ in [4] is replaced by $\big(\log(n)+\log\log(\Phi)+Q\big)$ here.) We note that [4] also give bounds independent of spread for a subclass of smooth kernels that we do not provide here.

Acknowledgements

We thank Arturs Backurs for presenting [4] at the Purdue Theory Seminar in August, 2019. We thank Chandra Chekuri for his helpful feedback in the writing of this work. We thank the anonymous reviewers for their helpful comments.

References

[1] Dimitris Achlioptas. "Database-friendly random projections". In: *Proceedings of the Twentieth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, May 21-23, 2001, Santa Barbara, California, USA*. Ed. by Peter Buneman. ACM, 2001.

[2] Josh Alman, Timothy Chu, Aaron Schild, and Zhao Song. "Algorithms and Hardness for Linear Algebra on Geometric Graphs". In: *61st IEEE Annual Symposium on Foundations of Computer Science, FOCS 2020*. 2020.

[3] Sanjeev Arora. "Polynomial Time Approximation Schemes for Euclidean Traveling Salesman and other Geometric Problems". In: *J. ACM* 45.5 (1998), pp. 753–782. URL: `https://doi.org/10.1145/290179.290180`. Preliminary version in FOCS, 1996.

[4] Arturs Backurs, Moses Charikar, Piotr Indyk, and Paris Siminelakis. "Efficient Density Evaluation for Smooth Kernels". In: *59th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2018, Paris, France, October 7-9, 2018*. 2018, pp. 615–626.

[5] Arturs Backurs, Piotr Indyk, and Ludwig Schmidt. "On the Fine-Grained Complexity of Empirical Risk Minimization: Kernel Methods and Neural Networks". In: *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*. Ed. by Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett. 2017, pp. 4308–4318. URL: `http://papers.nips.cc/paper/7018-on-the-fine-grained-complexity-of-empirical-risk-minimization-kernel-methods-and-neural-networks`.

[6] Arturs Backurs, Piotr Indyk, and Tal Wagner. "Space and Time Efficient Kernel Density Estimation in High Dimensions". In: *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, 8-14 December 2019, Vancouver, BC, Canada*. Ed. by Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett. 2019, pp. 15773–15782. URL: `http://papers.nips.cc/paper/9709-space-and-time-efficient-kernel-density-estimation-in-high-dimensions`.

[7] Kfir Barhum, Oded Goldreich, and Adi Shraibman. “On Approximating the Average Distance Between Points”. In: *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, 10th International Workshop, APPROX 2007, and 11th International Workshop, RANDOM 2007, Princeton, NJ, USA, August 20-22, 2007, Proceedings*. Ed. by Moses Charikar, Klaus Jansen, Omer Reingold, and José D. P. Rolim. Vol. 4627. Lecture Notes in Computer Science. Springer, 2007, pp. 296–310.

[8] Joshua D. Batson, Daniel A. Spielman, and Nikhil Srivastava. “Twice-Ramanujan Sparsifiers”. In: *SIAM J. Comput.* 41.6 (2012). Preliminary version in STOC, 2009, pp. 1704–1721.

[9] Rick Beatson and Leslie Greengard. “A short course on fast multipole methods”. English (US). In: *Wavelets, multilevel methods, and elliptic PDEs*. Ed. by M. Ainsworth, J. Levesley, M. Marletta, and W. A. Light. Numerical Mathematics and Scientific Computation. Oxford University Press, 1997. Chap. 1, pp. 1–37.

[10] Mikhail Belkin and Partha Niyogi. “Laplacian Eigenmaps for Dimensionality Reduction and Data Representation”. In: *Neural Comput.* 15.6 (2003), pp. 1373–1396. Preliminary version in NIPS, 2001.

[11] Mark de Berg, Otfried Cheong, Marc J. van Kreveld, and Mark H. Overmars. *Computational geometry: algorithms and applications, 3rd Edition*. Springer, 2008.

[12] Martin D. Buhmann. *Radial Basis Functions: Theory and Implementations*. Cambridge Monographs on Applied and Computational Mathematics. Cambridge University Press, 2003.

[13] Paul B. Callahan and S. Rao Kosaraju. “A Decomposition of Multidimensional Point Sets with Applications to k-Nearest-Neighbors and n-Body Potential Fields”. In: *J. ACM* 42.1 (1995). Preliminary version in STOC, 1992, pp. 67–90.

[14] Moses Charikar and Paris Siminelakis. “Hashing-Based-Estimators for Kernel Density in High Dimensions”. In: *58th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2017, Berkeley, CA, USA, October 15-17, 2017*. 2017, pp. 1032–1043.

[15] Moses Charikar and Paris Siminelakis. “Multi-resolution Hashing for Fast Pairwise Summations”. In: *60th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2019, Baltimore, Maryland, USA, November 9-12, 2019*. Ed. by David Zuckerman. IEEE Computer Society, 2019, pp. 769–792.

[16] Shiri Chechik, Edith Cohen, and Haim Kaplan. “Average Distance Queries through Weighted Samples in Graphs and Metric Spaces: High Scalability with Tight Statistical Guarantees”. In: *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2015, August 24-26, 2015, Princeton, NJ, USA*. Ed. by Naveen Garg, Klaus Jansen, Anup Rao, and José D. P. Rolim. Vol. 40. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2015, pp. 659–679.

[17] F. R. K. Chung. *Spectral Graph Theory*. American Mathematical Society, 1997.

[18] Artur Czumaj and Christian Sohler. “Sublinear-time Algorithms”. In: *Property Testing - Current Research and Surveys*. Ed. by Oded Goldreich. Vol. 6390. Lecture Notes in Computer Science. Springer, 2010, pp. 41–64.

[19] Anirban Dasgupta, Ravi Kumar, and Tamás Sarlós. “A sparse Johnson–Lindenstrauss transform”. In: *Proceedings of the 42nd ACM Symposium on Theory of Computing, STOC 2010, Cambridge, Massachusetts, USA, 5-8 June 2010*. Ed. by Leonard J. Schulman. ACM, 2010, pp. 341–350.

[20] Hossein Esfandiari and Michael Mitzenmacher. “Metric Sublinear Algorithms via Linear Sampling”. In: *59th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2018, Paris, France, October 7-9, 2018*. Ed. by Mikkel Thorup. IEEE Computer Society, 2018, pp. 11–22.

[21] Thomas Gärtner, Peter A. Flach, Adam Kowalczyk, and Alexander J. Smola. “Multi-Instance Kernels”. In: *Machine Learning, Proceedings of the Nineteenth International Conference (ICML 2002), University of New South Wales, Sydney, Australia, July 8-12, 2002*. Ed. by Claude Sammut and Achim G. Hoffmann. Morgan Kaufmann, 2002, pp. 179–186.

[22] Bogdan Georgescu, Ilan Shimshoni, and Peter Meer. “Mean Shift Based Clustering in High Dimensions: A Texture Classification Example”. In: *9th IEEE International Conference on Computer Vision (ICCV 2003), 14-17 October 2003, Nice, France*. IEEE Computer Society, 2003, pp. 456–463.

[23] Leslie Greengard. “Fast Algorithms for Classical Physics”. In: *Science* 265.5174 (1994), pp. 909–914. eprint: `https://science.sciencemag.org/content/265/5174/909.full.pdf`. URL: `https://science.sciencemag.org/content/265/5174/909`.

[24] Leslie Greengard and Vladimir Rokhlin Jr. “A fast algorithm for particle simulations”. In: *Journal of Computational Physics* 73.2 (1987), pp. 325 –348. URL: `http://www.sciencedirect.com/science/article/pii/0021999187901409`.

[25] Sariel Har-Peled. *Geometric Approximation Algorithms*. USA: American Mathematical Society, 2011.

[26] Sariel Har-Peled and Manor Mendel. “Fast Construction of Nets in Low-Dimensional Metrics and Their Applications”. In: *SIAM J. Comput.* 35.5 (2006). Preliminary version in SOCG, 2005, pp. 1148–1184. URL: `https://doi.org/10.1137/S0097539704446281`.

[27] Michael P. Holmes, Alexander G. Gray, and Charles Lee Isbell Jr. “Ultrafast Monte Carlo for Statistical Summations”. In: *Advances in Neural Information Processing Systems 20, Proceedings of the Twenty-First Annual Conference on Neural Information Processing Systems, Vancouver, British Columbia, Canada, December 3-6, 2007*. Ed. by John C. Platt, Daphne Koller, Yoram Singer, and Sam T. Roweis. Curran Associates, Inc., 2007, pp. 673–680.

[28] Piotr Indyk. “A Sublinear Time Approximation Scheme for Clustering in Metric Spaces”. In: *40th Annual Symposium on Foundations of Computer Science, FOCS ’99, 17-18 October, 1999, New York, NY, USA*. IEEE Computer Society, 1999, pp. 154–159.

[29] Piotr Indyk. “Sublinear Time Algorithms for Metric Space Problems”. In: *Proceedings of the Thirty-First Annual ACM Symposium on Theory of Computing, May 1-4, 1999, Atlanta, Georgia, USA*. Ed. by Jeffrey Scott Vitter, Lawrence L. Larmore, and Frank Thomson Leighton. ACM, 1999, pp. 428–434.

[30] Piotr Indyk and Rajeev Motwani. “Approximate Nearest Neighbors: Towards Removing the Curse of Dimensionality”. In: *Proceedings of the Thirtieth Annual ACM Symposium on the Theory of Computing, Dallas, Texas, USA, May 23-26, 1998*. Ed. by Jeffrey Scott Vitter. ACM, 1998, pp. 604–613.

[31] William Johnson and Joram Lindenstrauss. “Extensions of Lipschitz maps into a Hilbert space”. In: *Contemporary Mathematics* 26 (Jan. 1984), pp. 189–206.

[32] Daniel M. Kane and Jelani Nelson. “A Derandomized Sparse Johnson-Lindenstrauss Transform”. In: *CoRR* abs/1006.3585 (2010). arXiv: `1006.3585`. URL: `http://arxiv.org/abs/1006.3585`.

[33] Daniel M. Kane and Jelani Nelson. “Sparser Johnson-Lindenstrauss Transforms”. In: *J. ACM* 61.1 (2014), 4:1–4:23. Preliminary version in SODA, 2012.

[34] Ravi Kannan and Santosh S. Vempala. “Spectral Algorithms”. In: *Foundations and Trends in Theoretical Computer Science* 4.3-4 (2009), pp. 157–288.

[35] Yin Tat Lee and He Sun. “Constructing Linear-Sized Spectral Sparsification in Almost-Linear Time”. In: *SIAM J. Comput.* 47.6 (2018). Preliminary version in FOCS, 2017, pp. 2315–2336.

[36] Joseph S. B. Mitchell. “Guillotine Subdivisions Approximate Polygonal Subdivisions: A Simple Polynomial-Time Approximation Scheme for Geometric TSP, k-MST, and Related Problems”. In: *SIAM J. Comput.* 28.4 (1999), pp. 1298–1309. URL: `https://doi.org/10.1137/S0097539796309764`.

[37] Andrew Y. Ng, Michael I. Jordan, and Yair Weiss. "On Spectral Clustering: Analysis and an algorithm". In: *Advances in Neural Information Processing Systems 14 [Neural Information Processing Systems: Natural and Synthetic, NIPS 2001, December 3-8, 2001, Vancouver, British Columbia, Canada]*. Ed. by Thomas G. Dietterich, Suzanna Becker, and Zoubin Ghahramani. MIT Press, 2001, pp. 849–856. URL: http://papers.nips.cc/paper/2092-on-spectral-clustering-analysis-and-an-algorithm.

[38] Alessandro Rinaldo and Larry Wasserman. "Generalized density clustering". In: *Ann. Statist.* 38.5 (Oct. 2010), pp. 2678–2722.

[39] Jeffrey S. Salowe. "Constructing multidimensional spanner graphs". In: *Int. J. Comput. Geom. Appl.* 1.2 (1991), pp. 99–107.

[40] Erich Schubert, Arthur Zimek, and Hans-Peter Kriegel. "Generalized Outlier Detection with Flexible Kernel Density Estimates". In: *Proceedings of the 2014 SIAM International Conference on Data Mining, Philadelphia, Pennsylvania, USA, April 24-26, 2014*. Ed. by Mohammed Javeed Zaki, Zoran Obradovic, Pang-Ning Tan, Arindam Banerjee, Chandrika Kamath, and Srinivasan Parthasarathy. SIAM, 2014, pp. 542–550.

[41] Paris Siminelakis, Kexin Rong, Peter Bailis, Moses Charikar, and Philip Levis. "Rehashing Kernel Evaluation in High Dimensions". In: *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*. Ed. by Kamalika Chaudhuri and Ruslan Salakhutdinov. Vol. 97. Proceedings of Machine Learning Research. PMLR, 2019, pp. 5789–5798.

[42] Daniel A. Spielman. "Spectral and Algebraic Graph Theory". Draft. 2019. URL: https://www.cs.yale.edu/homes/spielman/sagt/sagt.pdf.

[43] Daniel A. Spielman and Nikhil Srivastava. "Graph Sparsification by Effective Resistances". In: *SIAM J. Comput.* 40.6 (2011). Preliminary version in STOC, 2008, pp. 1913–1926.

[44] Zoltán Szabó, Bharath K. Sriperumbudur, Barnabás Póczos, and Arthur Gretton. "Learning Theory for Distribution Regression". In: *J. Mach. Learn. Res.* 17 (2016), 152:1–152:40. URL: http://jmlr.org/papers/v17/14-510.html.

[45] Luca Trevisan. "Lecture Notes on Expansion, Sparsest Cut, and Spectral Graph Theory". Accompanying courses taught at UC Berkeley in 2011, 2014, and 2016. 2011.

[46] Joel A. Tropp. "User-Friendly Tail Bounds for Sums of Random Matrices". In: *Found. Comput. Math.* 12.4 (2012), pp. 389–434. URL: https://doi.org/10.1007/s10208-011-9099-z.

[47] Pravin M. Vaidya. "A sparse Graph Almost as Good as the Complete Graph on Points in K Dimensions". In: *Discret. Comput. Geom.* 6 (1991), pp. 369–381.