EPFL

Thèse n° 8993

Sketches, metrics and fast algorithms

Présentée le 9 novembre 2022

Faculté informatique et communications
Laboratoire de théorie du calcul 4
Programme doctoral en informatique et communications

pour l'obtention du grade de Docteur ès Sciences

par

Navid NOURI

Acceptée sur proposition du jury

Prof. M. T. Göös, président du jury
Prof. M. Kapralov, directeur de thèse
Prof. M. Thorup, rapporteur
Prof. A. Andoni, rapporteur
Prof. O. Svensson, rapporteur

École
polytechnique
fédérale
de Lausanne

2022

To my parents...

Acknowledgements

I have many people to thank for making my five-year stay at EPFL memorable. The following is certainly an incomplete attempt.

I was very fortunate to have Michael as my advisor and I would like to thank him for his support, guidance and for introducing me to many great people. It would not have been possible for me to complete my PhD studies successfully without his mentorship. I want to thank the jury members of my thesis defense, Mikkel Thorup, Ola Svensson, Alexandr Andoni and Mika Göös for agreeing to be on the committee and the discussions we had during the oral exam. I am also grateful to many of my collaborators for stimulating discussions. More specifically, I want to thank Moses Charikar, for showing me the right attitude towards research. Moreover, I want to thank Amir Zandieh, Jakab Tardos and Arnold Filtser, for their research passion, hard work and dedication to get things done in our research projects.

Another special thanks to European Research Council, for financially supporting our research during this period.[1]

I also want to thank Pauline Raffestin who always kindly helped me deal with all administrative processes in the smoothest way possible.

I am grateful to Credit Suisse and JPMorgan Chase & Co. for the internship opportunities. More specifically, I want to thank Elham Zargari, for being a very supportive manager during my internship at Credit Suisse. And I want to express my gratefulness to Eleonora Kreacic and Vamsi Krishna Potluru for the awesome experience I had in the AI research team at JPMorgan.

My special appreciation goes to my dear friends. In particular, I want to thank Omid (Mashinchian), Amir and Khashayar for all the fun and enjoyable moments we had together. I also want to thank my long-distance friends, Morteza, Omid (Sadeghi) and Saghar for our long and yet enjoyable discussions about almost everything.

Finally, and most importantly, I want to thank my parents, for their endless support, never-ending sacrifices and their understanding throughout this journey. They were always the most caring and kind parents imaginable. And a special thanks to my lovely girlfriend, Darya, who was supportive and understanding during all these years.

[1] I was supported by ERC Starting Grant 759471.

Abstract

As it has become easier and cheaper to collect big datasets in the last few decades, designing efficient and low-cost algorithms for these datasets has attracted unprecedented attention. However, in most applications, even storing datasets as acquired has become extremely costly and inefficient, which motivates the study of sublinear algorithms. This thesis focuses on studying two fundamental graph problems in the sublinear regime. Furthermore, it presents a fast kernel density estimation algorithm and data structure.

The first part of this thesis focuses on graph spectral sparsification in dynamic streams. Our algorithm achieves almost optimal runtime and space simultaneously in a single pass. Our method is based on a novel bucketing scheme that enables us to recover high effective resistance edges faster. This contribution presents a novel approach to the effective resistance embedding of the graph, using locality-sensitive hash functions, with possible further future applications.

The second part of this thesis presents spanner construction results in the dynamic streams and the simultaneous communication models. First, we show how one can construct a $\widetilde{O}(n^{2/3})$-spanner using the above-mentioned almost-optimal single-pass spectral sparsifier, resulting in the first single-pass algorithm for non-trivial spanner construction in the literature. Then, we generalize this result to constructing $\widetilde{O}(n^{2/3(1-\alpha)})$-spanners using $\widetilde{O}(n^{1+\alpha})$ space for any $\alpha \in [0,1]$, providing a smooth trade-off between distortion and memory complexity. Moreover, we study the simultaneous communication model and propose a novel protocol with low per player information. Also, we show how one can leverage more rounds of communication in this setting to achieve better distortion guarantees.

Finally, in the third part of this thesis, we study the kernel density estimation problem. In this problem, given a kernel function, an input dataset imposes a kernel density on the space. The goal is to design fast and memory-efficient data structures that can output approximations to the kernel density at queried points. This thesis presents a data structure based on the classical near neighbor search and locality-sensitive hashing techniques that improves or matches the query time and space complexity for radial kernels considered in the literature. The approach is based on an implementation of (approximate) importance sampling for each distance range and then using near neighbor search algorithms to recover points from these distance ranges. Later, we show how to improve the runtime, using recent advances in the data-dependent near neighbor search data structures, for a class of radial kernels that includes the Gaussian kernel.

Keywords: Sublinear Algorithms, Spectral Graph Theory, Graph Sparsification, Spanners, Dynamic Streams, Simultaneous Communication Model, Kernel Density Estimation.

Zusammenfassung

Da es in den letzten Jahrzehnten einfacher und billiger geworden ist, große Datenmengen zu sammeln, hat die Entwicklung effizienter und kostengünstiger Algorithmen für diese Datenmengen beispiellose Aufmerksamkeit auf sich gezogen. In den meisten Anwendungen ist jedoch selbst das Speichern von erfassten Datensätzen extrem kostspielig und ineffizient geworden, was die Studie sublinearer Algorithmen motiviert. Diese Dissertation konzentriert sich auf die Untersuchung zweier grundlegender Graphenproblemen im sublinearen Regime. Darüber hinaus stellt sie einen schnellen Algorithmus und eine Datenstruktur zur Schätzung der Kerndichte bereit.

Der erste Teil dieser Arbeit konzentriert sich auf die Sparsifizierung von Graphenspektralen in dynamischen Streams. Unser Algorithmus erreicht gleichzeitig in einem einzigen Durchgang eine nahezu optimale Laufzeit und Speicherplatz. Unsere Methode basiert auf einem neuartigen Bucket-Schema, das es uns ermöglicht, hocheffektive Resistenz-Kanten schneller wiederherzustellen. Dieser Beitrag präsentiert einen neuartigen Ansatz zur effektiven Resistenz-Einbettung des Graphen unter Verwendung ortsabhängiger Hash- Funktionen mit möglichen weiteren zukünftigen Anwendungen.

Der zweite Teil dieser Arbeit präsentiert Ergebnisse der Spanner-Konstruktion in den dynamischen Streams und den simultanen Kommunikationsmodellen. Zuerst zeigen wir, wie man einen $\widetilde{O}(n^{2/3})$-Spanner unter Verwendung des oben erwähnten fast optimalen Single-Pass-Spektralsparsifiers konstruieren kann, was zum ersten Single-Pass-Algorithmus für nicht-triviale Spanner-Konstruktionen in der Literatur führt. Dann verallgemeinern wir dieses Ergebnis zum Konstruieren von $\widetilde{O}(n^{2/3(1-\alpha)})$-Schlüsseln unter Verwendung von $\widetilde{O}(n^{1+\alpha})$-Raum für jedes $\alpha \in [0,1]$, was zum einen glatten Trade off zwischen Verzerrung und Speicherkomplexität führt. Darüber hinaus untersuchen wir das simultane Kommunikationsmodell und schlagen ein neuartiges Protokoll mit geringen Informationen pro Spieler vor. Außerdem zeigen wir, wie man in dieser Umgebung mehr Kommunikationsrunden nutzen kann, um bessere Verzerrungsgarantien zu erreichen.

Schließlich untersuchen wir im dritten Teil dieser Arbeit das Kerndichteschätzungsproblem. Bei diesem Problem erlegt ein Eingabedatensatz bei gegebener Kern-Funktion dem Raum eine Kerndichte auf. Ziel ist es, schnelle und speichereffiziente Datenstrukturen zu entwerfen, die Annäherungen an die Kerndichte an abgefragten Punkten ausgeben können. Diese Arbeit stellt eine Datenstruktur vor, die auf der klassischen Suche nach nahen Nachbarn und lokalitätssensitiven Hashing-Techniken basiert, die die in der Literatur betrachtete Abfragezeit und Raumkomplexität für radiale Kern verbessert oder anpasst. Der Ansatz basiert auf einer Implementierung einer (ungefähren) Wichtigkeitsabtastung für jeden

Entfernungsbereich und dann der Verwendung von Suchalgorithmen für nahe Nachbarn, um Punkte aus diesen Entfernungsbereichen wiederherzustellen. Später zeigen wir, wie die Laufzeit verbessert werden kann, indem wir die jüngsten Fortschritte in den datenabhängigen Datenstrukturen für die Suche nach nahen Nachbarn für eine Klasse von radialen Kernen verwenden, die den Gaußschen Kern enthalten.

Zusammenfassung Schlüsselwörter: Sublineare Algorithmen, Spectrale Graphentheorie, Graph Sparsifizierung, Spanners, dynamische Streams, Simultaten Kommunikationsmodelle, Kerndichteschätzung.

Contents

Introduction

Modern daily life is deeply affected by algorithms and their products, ranging from social media networks and search engines to even the simplest smartphone applications. Recently, collecting massive datasets has become relatively cheap, and data analysis techniques are almost common knowledge for engineers and scientists in almost every area. Furthermore, the growth of dataset volumes is crippling existing algorithms and demands more time and memory-efficient algorithms and data structures.

One commonly used solution for the challenges mentioned above is to trade accuracy for less cost (e.g., time, memory usage), using approximation algorithms, where an approximate output is acceptable. In most practical settings, as there are already numerical errors imposed on the final result, using approximate algorithms that run comparably faster than previously known algorithms is an appealing alternative.

Moreover, motivated by the fact that large graphs are generally expected to be dynamic, the streaming model for graphs has been introduced. More specifically, in the dynamic streaming model for graphs, the input is a stream of edge insertions or deletions. The algorithm is supposed to update its limited memory accordingly to output an answer for a predefined problem afterward. Ideally, one expects the following properties for a dynamic streaming algorithm. First, the memory usage needs to be (almost) linear in the output size. Second, the update time of the memory state after each update is expected to be (almost) constant time. Finally, the algorithm generates the output in time (almost) linear in the size of the output. Graph sketching is essentially the only general purpose tool to handle dynamic streams for graphs in a space-efficient manner, first introduced by the seminal work of Ahn, Guha, and McGregor [1]. Given a graph G with n vertices and edge-vertex incidence matrix B, a random projection ΠB, where $\Pi \in \mathbb{R}^{d \times \binom{n}{2}}$ and $d \ll n$, is called a (linear) sketch of G. Note that the sketches are not limited to this definition. However, this thesis focuses on this specific family of sketches.

The increasing need for designing parallel algorithms motivates the study of a less-discovered model called the simultaneous communication model. In the simultaneous communication model, each vertex in the graph is considered a machine with a list of its neighbors, and all machines have access to a source of shared randomness. In each round, all the nodes are allowed to broadcast a limited-length message. It is somewhat apparent that sketching the edge-vertex incidence matrix naturally yields sketches with low communication in the simultaneous communication model.

In this thesis, we study two graph approximation notions, namely spectral sparsification and spanner construction, in the computation models defined above. The rest of the thesis is focused on Kernel Density Estimation problem. Below, we introduce the main questions investigated in this thesis.

Graph spectral sparsification in dynamic streams. One major question in graph theory is to sparsify dense graphs while preserving, in some cases approximately, some predetermined properties, such as cut sizes and spectral properties. As a result, graph sparsification algorithms have gained massive attention in recent years. The first attempt was to approximately preserve cut sizes in graphs [2, 3], while sparsifying the graph to $\widetilde{O}(n)$ number of edges. This line of research later continued with a more general approximation guarantee called spectral approximation by the seminal work of Spielman and Teng [4]. Below, we informally define spectral sparsifiers.

Definition 0.0.1 (Informal)**.** *A graph $\widetilde{G}$ is a spectral sparsifier of graph G if it satisfies the following conditions. First, for any real-valued vector x, $x^T L_G x \approx x^T L_{\widetilde{G}} x$, where L_G and $L_{\widetilde{G}}$ are the Laplacian matrices of G and $\widetilde{G}$, respectively. Second, $\widetilde{G}$ has fewer edges than G.*

Spectral sparsification has been studied extensively for the setting that the graph is given as input; see the survey [5] for a comprehensive set of pioneering works. Recently, with the extensive need for algorithms with low space complexity on dynamic streams, [6, 7] studied the spectral sparsification problem in the streaming model. However, these results suffered from overhead in the memory (and/or) overhead in the runtime.

Spanners in dynamic streams and in the simultaneous communication model. Another challenging question in designing graph algorithms in dynamic streams is to design an algorithm that outputs a spanner of the input graph. Below, we informally define spanners.

Definition 0.0.2 (Informal)**.** *A spanner of graph G is a subgraph H of the input graph that preserves the shortest path metric between pair of vertices up to some error factor.*

In a single pass dynamic stream, the best-known result in the literature that uses almost linear space constructs a spanning tree, which cannot guarantee a better worst-case distortion than $(n-1)$-stretch. Thus, one main open question was whether it is possible to construct a spanner with a $o(n)$ distortion using almost linear space. Another undiscovered question is to design algorithms in the simultaneous communication model. In this model, as explained above, initially, each node knows its list of neighbors in the graph. Then, in each round, the nodes post a short message on a shared board. The goal is to be able to extract a correct output with a high probability from the content published on the board. Although there are inherent similarities between this model and the sketching ideas used in streaming models, we discuss the fundamental differences in Chapter 2.

Kernel Density Estimation. Kernel Density Estimation (KDE) is one of the critical problems in machine learning, statistics, and data analysis. Below, we informally define the kernel density estimation problem discussed in this thesis.

Definition 0.0.3 (Informal)**.** *Given a kernel function $K(\cdot,\cdot)$ and a set of data points P in a Euclidean space, a kernel density estimation algorithm prepares an efficient data structure so that when a query point q is received, the Kernel Density (KD) at the query point, i.e.,*

$$\frac{1}{|P|}\sum_{p\in P} K(p,q), \tag{1}$$

can be approximated using a relatively fast procedure.

This problem has received much attention among researchers in recent years; however, in most cases, the algorithms are suffering from the curse of dimensionality. A recent line of work [8, 9], studied sublinear algorithms for this problem in high dimensions. The dominant idea is implementing a coarse importance sampling procedure using locality sensitive hashing (LSH). The intuition is based on the fact that as close points have a higher contribution to the KD at the query point, we can leverage LSH functions to implement a rough version of importance sampling. Although this approach paved the way to a better understanding of the problem and breakthrough results, the variance analysis of this approach remained cumbersome. Therefore, the possibility of designing a simple and easy-to-analyze algorithm for this problem that can improve upon prior results remained unclear in the literature. Furthermore, the implications of new advanced data-dependent LSH techniques for this problem were undiscovered.

Contribution overview

This thesis focuses on designing fast and space-efficient algorithms for the two graph approximation notions mentioned above, spectral sparsification and spanner construction, as well as the kernel density estimation problem. We leverage appropriate graph sketches in the dynamic streaming model and the simultaneous communication model to achieve low memory cost data structures and fast algorithms for the graph approximation problems. On the other hand, we present a sublinear time and space data structure for the kernel density estimation problem. The following paragraphs discuss the summary of our contributions in more detail.

In the first part of this thesis, using a novel approach based on bucketing vertices in the effective resistance metric, we present an algorithm that achieves almost optimal runtime and space complexity simultaneously for the graph sparsification problem in the dynamic streaming model. In general, one can reduce the sparsification problem to the problem of recovering edges with high effective resistance using the sketch of the graph. This problem has been solved in [7] by brute-forcing over all pairs of vertices, i.e., using $\widetilde{O}(n^2)$ test vectors. On the other hand, this thesis shows how this problem can be solved using $\widetilde{O}(n)$ test vectors by leveraging a bucketing approach, using an LSH function on the effective resistance embedding of the graph. This is to ensure that only close points in the effective resistance metric can fall into the same buckets and use this fact to design an almost linear-sized set of pairs of vertices to recover all high effective resistance edges.

In the second part of this thesis, we show how the result of the first part can be used to construct $o(n)$-spanners in a single pass over a dynamic stream. More specifically, we show how one can obtain $\widetilde{O}(n^{2/3})$-spanners using spectral sparsifiers with $\widetilde{O}(n)$ space. This result improves the best-known (trivial)

approach of constructing a spanning tree that incurs a worst-case $\Omega(n)$ distortion to the shortest path metric. Moreover, we show how one can trade more space for a better distortion, namely our algorithm generates $\widetilde{O}(n^{2/3(1-\alpha)})$-spanners using $\widetilde{O}(n^{1+\alpha})$ space, in one pass over the dynamic stream. Later, we also investigate this problem in the simultaneous communication model. We show a trade-off between the number of rounds nodes can broadcast their messages and the distortion.

In the last part of this thesis, we present sublinear algorithms to tackle the kernel density estimation problem in high dimensions. First, we present a simple and easy-to-analyze algorithm by implementing importance sampling and leveraging the properties of data-independent Andoni-Indyk LSH [10]. This algorithm improves upon or up to polylogarithmic factors matches the best-known results for radial kernels such as the Gaussian and exponential kernels. As opposed to the best-known algorithms in the literature [8, 9], we do not use an LSH function as a tool to implement a coarse importance sampling. Instead, we leverage the fact that a certain amount of KD at some query point imposes an upper bound on the distribution of points at each distance, which we call density constraints. Furthermore, noting that points from the same distance contribute the same amount, in order to sample points from a specific distance range, with a probability proportional to their contribution (i.e., importance sampling), one can first subsample the whole dataset with that rate, and then use LSH function to recover points in that distance range, for any given query point. The crucial point is that density constraints limit the number of points that will end up in the same bucket as the query in the LSH function, which results in fast query time.

Additionally, we improve our algorithm by utilizing new data-dependent LSH data structures [11, 12, 13]. Our data-dependent algorithm not only improves the result of the first part for a class of kernels that includes the Gaussian kernel but also shed light on understanding the behavior of data-dependent LSH functions when there are constraints over the distribution of the input dataset.

Organization

The following three chapters discuss graph sparsification in dynamic streams, spanner construction in dynamic streams and the simultaneous communication model, and kernel density estimation problem, respectively. The results of each chapter have been presented in a self-contained way and can be read independently. In addition, each chapter has its introduction covering a more comprehensive set of prior work and a thorough explanation of techniques and contributions.

1 Dynamic Streaming Spectral Sparsification in Nearly Linear Time and Space

This chapter is based on a joint work with Michael Kapralov, Aida Mousavifar, Cameron Musco, Christopher Musco, Aaron Sidford and Jakab Tardoss. It has been accepted to the ACM-SIAM Symposium on Discrete Algorithms [14, SODA].

1.1 Introduction

Graph sketching, i.e. constructing small space summaries for graphs using linear measurements, has received much attention since the work of Ahn, Guha and McGregor [1] gave a linear sketching primitive for graph connectivity with optimal $O(n\log^3 n)$ space complexity [15]. A key application of linear sketching has been to design small space algorithms for processing *dynamic graph streams*, where edges can be both inserted and deleted, although the graph sketching paradigm has been shown very powerful in many other areas such as distributed algorithms and dynamic algorithms (we refer the reader to the survey [16] for more on applications of graph sketching). Furthermore, it is known that linear sketching is essentially a universal approach to designing dynamic streaming algorithms [17], and yields distributed protocols for graph processing with low communication. Sketching solutions have been recently constructed for many graph problems, including spanning forest computation [1], cut and spectral sparsifiers [18, 7], spanner construction [18, 19], matching and matching size approximation [20, 21], sketching the Laplacian [22, 23] and many other problems. The focus of our work is on *oblivious* sketches for approximating spectral structure of graphs with optimally fast recovery. A sketch is called *oblivious* if its distribution is independent of the input – such sketches yield efficient *single pass* dynamic streaming algorithms for sparsification. We now outline the main ideas involved in previous works on this and related problems, and highlight the main challenges in designing a solution that achieves both linear space and time.

Oblivious linear sketches with nearly optimal $n\log^{O(1)} n$ have been obtained for the related problems of constructing a spanning forest of the input graph [1], the problem of constructing *cut sparsifiers* of graphs [24] and for the spectral sparsification problem itself [7]. In the former two cases the core of the problem is to design a sketch that allows recovery of edges that cross *small cuts* in the input graph, and the problem is resolved by applying ℓ_0-sampling(see, e.g., [25, 26, 27]), and more generally exact (i.e., ℓ_0) sparse recovery techniques on the edge incidence matrix $B \in \mathbb{R}^{\binom{n}{2}\times n}$ of the input graph: one designs a sketching matrix $S \in \mathbb{R}^{\log^{O(1)} n\times\binom{n}{2}}$ and maintains $S\cdot B \in \mathbb{R}^{\log^{O(1)} n\times n}$ throughout the stream. A natural

recovery primitive that follows Boruvka's algorithm for the MST problem then yields a nearly linear time recovery scheme. Specifically, to recover a spanning tree one repeatedly samples outgoing edges out of every vertex of the graph and contracts resulting connected components into supernodes, halving the number of connected components in every round. Surprisingly, a sketch of the original graph suffices for sampling edges that go across connected components in graphs that arise through the contraction process, yielding a spanning forest in $O(\log n)$ rounds and using $n\log^{O(1)} n$ bits of space.

The situation with spectral sparsifiers is very different: edges critical to obtaining a spectral approximation do not necessarily cross small cuts in the graph. Instead, 'important edges' are those that have large effective resistance, i.e can be made 'heavy' in the ℓ_2 sense in an appropriate linear combination of the columns of the edge incidence matrix B. This observation was used in [7] to design a sketch with nearly optimal $n\log^{O(1)} n$ space complexity, but the recovery of the sparsifier was brute-force and ran in $\Omega(n^2)$ time: one had to iterate over all potential edges and test whether they are in the graph and have 'high' effective resistance. Approaches based on relating effective resistances to inverse connectivity have been proposed [6], but these result in suboptimal $\Omega(n^{5/3})$ space complexity. In a very recent work [28] a subset of the authors proposed an algorithm with $n^{1.4+o(1)}$ space and runtime complexity, but no approach that yields optimal space and runtime was known previously.

A key reason why previously known sketching techniques for reconstructing spectral approximations to graphs failed to achieve nearly linear runtime is exactly the lack of simple 'local' (akin to Boruvka's algorithm) technique for recovering heavy edges. The main contribution of this paper is such a technique: we propose a bucketing technique based on ball carving in (an approximation to) the effective resistance metric that recovers appropriately heavy effective resistance edges by routing flows between source-sink pairs that belong to the same bucket. This ensures that the recovery process is more 'localized', and results in a nearly linear time algorithm.

Our result. Formally, we consider the problem of constructing *spectral sparsifiers* [29, 30] of graphs presented as a dynamic stream of edges: given a graph $G = (V, E)$ presented as a dynamic stream of edge insertions and deletions and a precision parameter $\epsilon \in (0,1)$, our algorithm outputs a graph G' such that

$$(1-\epsilon)L \preceq L' \preceq (1+\epsilon)L,$$

where L is the Laplacian of G, L' is the Laplacian of G' and $\prec$ stands for the positive semidefinite ordering of matrices.

Our main result is a linear sketching algorithm that compresses a graph with n vertices to a $n\log^{O(1)} n$-bit representation that allows $\log^{O(1)} n$-time updates, and from which a spectral approximation can be recovered in $n\log^{O(1)} n$ time. Thus, our result achieve both optimal space and time complexity simultaneously.

Theorem 1.1.1 (Near Optimal Streaming Spectral Sparsification)**.** *There exists an algorithm such that for any $\epsilon \in (0,1)$, processes a list of edge insertions and deletions for an unweighted graph G in a single pass and maintains a set of linear sketches of this input in $O(\epsilon^{-2} n\log^{O(1)} n)$ space. From these sketches, it recovers in $O(\epsilon^{-2} n\log^{O(1)} n)$ time, with high probability, a weighted subgraph H with $O(\epsilon^{-2} n\log n)$ edges, such that H is a $(1\pm\epsilon)$-spectral sparsifier of G.*

Our result in Theorem 1.1.1 can be thought of as the first efficient 'ℓ_2-graph sketching' result, using an analogy to compressed sensing recovery guarantees. It is interesting to note that compressed sensing primitives that allow recovery in time nearly linear in sketch size (which is exactly what our algorithm achieves for the sparsification problem) usually operate by hashing the input vector into buckets so as to isolate dominant entries, which can then be recovered efficiently. The main contribution of our work is giving a 'bucketing scheme' for graphs that allows for nearly linear time recovery. As we show, the right 'bucketing scheme' for the spectral sparsification problem is a space partitioning scheme in the effective resistance metric.

Effective resistance, spectral sparsification, and random spanning trees. The *effective resistance metric* or *effective resistance distances* induced by an undirected graph plays a central role in spectral graph theory and has been at the heart of numerous algorithmic breakthroughs over the past decade. They are central to the to obtaining fast algorithms for constructing spectral sparsifiers [29, 31], spectral vertex sparsifiers [32], sparsifiers of the random walk Laplacian [33, 34], and subspace sparsifiers [35]. They have played a key role in many advances in solving Laplacian systems [4, 36, 37, 38, 39, 31, 32, 40] and are critical to the current fastest (weakly)-polynomial time algorithms for maximum flow and minimum cost flow in certain parameter regimes [41]. Given their utility, the computation of effective resistances has itself become an area of active research [23, 42].

In a line of work particularly relevant to this paper, the effective resistance metric has played an important role in obtaining faster algorithms for generating random spanning trees [43, 44, 45]. The result of [44] partitions the graph into clusters with bounded diameter in the effective resistance metric in order to speed up simulation of a random walk, whereas [45] proposed a more advanced version of this approach to achieve a nearly linear time simulation. While these results seem superficially related to ours, there does not seem to be any way of using spanning tree generation techniques for our purpose. The main reason is that the objective in spanning tree generation results is quite different from ours: there one would like to find a partition of the graph that in a sense minimizes the number times a random walk crosses cluster boundaries, which does not correspond to a way of recovering 'heavy' effective resistance edges in the graph. In particular, while in spanning tree generation algorithms the important parameter is the number of edges crossing the cuts generated by the partitioning, whereas it is easily seen that heavy effective resistance edges cannot be recovered from small cuts. Finally, the problem of partitioning graphs into low effective resistance diameter clusters has been studied recently in [46]. The focus of the latter work is on partitioning into *induced* expanders, and the results of [46] were an important tool in the work of [28] that achieved the previous best $n^{1.4+o(1)}$ space and runtime complexity for our problem. Our techniques in this paper take a different route and achieve optimal results.

Prior work. Streaming algorithms are well-studied with too many results to list and we refer the reader to [47, 16] for a survey of streaming algorithms. The idea of linear graph sketching was introduced in a seminal paper of Ahn, Guha, and McGregror [1], where a $O(\log n)$-pass sparsification algorithm for dynamic streams was presented (this result is for the weaker notion of cut sparsification due to [2, 3]). A single-pass algorithm for cut sparsification with nearly optimal $\widetilde{O}(\epsilon^{-2}n)$ space was given in [24], and extensions of the sketching approach of [1] to hypergraphs were presented in [48]. The more challenging

problem of computing a spectral sparsifier from a linear sketch was addressed in [6], who gives an $\tilde{O}(\epsilon^{-2}n^{5/3})$ space solution. An $\tilde{O}(\epsilon^{-2}n)$ space solution was obtained in [7] by more explicitly exploiting the connection between graph sketching and vector sparse recovery, at the expense of $\widetilde{O}(\epsilon^{-2}n^2)$ runtime. In a recent work [28] a subset of the authors gave a single pass algorithm with $\epsilon^{-2}n^{1.4+o(1)}$ space and runtime complexity.

We also mention that spectral sparsifiers have been studied in the insertion-only streaming model, where edges can only be added to G [49, 50, 51], and in a dynamic data structure model [52, 22, 23], where more space is allowed, but the algorithm must quickly output a sparsifier at every step of the stream. While these models are superficially similar to the dynamic streaming model, they seem to allow for different techniques, and in particular do not require linear sketching since they do not constrain the space used by the algorithm. The spectral sparsification problem on its own has received a lot of attention in the literature (e.g., [30, 53, 54, 55, 56, 57]. We refer the reader to the survey [5] for a more complete set of references.

1.2 Preliminaries

General Notation. Let $G = (V, E)$ be an unweighted undirected graph with n vertices and m edges. For any vertex $v \in V$, let $\chi_v \in \mathbb{R}^n$ be the indicator vector of v, with a one at position v and zeros elsewhere. Let $B_n \in \mathbb{R}^{\binom{n}{2}\times n}$ denote the vertex edge incidence matrix of an unweighted and undirected complete graph, where for any edge $e = (u, v) \in \binom{V}{2}$, $u \neq v$, its e'th row is equal to $\mathbf{b}_e := \mathbf{b}_{uv} := \chi_u - \chi_v$. Let $B \in \mathbb{R}^{\binom{n}{2}\times n}$ denote the vertex edge incidence matrix of $G = (V, E)$. B is obtained by zeroing out any rows of B_n corresponding to $(u, v) \notin E$.[1]

For weighted graph $G = (V, E, w)$, where $w : E \to \mathbb{R}_+$ denotes the edge weights, let $W \in \mathbb{R}_+^{\binom{n}{2}\times\binom{n}{2}}$ be the diagonal matrix of weights where $W(e, e) = w(e)$ for $e \in E$ and $W(e, e) = 0$ otherwise. Note that $L = B^\top WB = B_n^T WB_n$, is the Laplacian matrix of G. Let L^+ denote the Moore-Penrose pseudoinverse of L. Also, for a real valued variable s, we define $s^+ := \max\{0, s\}$. We also use the following folklore:

Fact 1.2.1. *For any Laplacian matrix L of an unweighted and undirected graph, its minimum nonzero eigenvalue is bounded from below by $\lambda_\ell = \frac{1}{8n^2}$ and its maximum eigenvalue is bounded from above by $\lambda_u = 2n$.*

Definition 1.2.1. *For any unweighted graph $G = (V, E)$ and any $\gamma \geq 0$, we define L_{G^γ}, as follows:*

$$L_{G^\gamma} = L_G + \gamma I.$$

This can be seen in the following way. One can think of G^γ as graph G plus some regularization term. In order to distinguish between edges of G and regularization term in G^γ, we let $B_{G^\gamma} = B \oplus \sqrt{\gamma} I$, where $B \oplus \sqrt{\gamma} I$ is the operation of appending rows of $\sqrt{\gamma} I$ to matrix B. One should note that $B_{G^\gamma}^\top B_{G^\gamma} = L_{G^\gamma}$. Also

[1] Note this is different then the possibly more standard definition of B as the $E \times V$ matrix with the rows not in the graph removed altogether.

for simplicity we define L_ℓ for any integer $\ell \in [0, d+1]$ as follows:

$$L_\ell = \begin{cases} L_G + \frac{\lambda_u}{2^\ell} I & \text{if } 0 \le \ell \le d \\ L_G & \text{if } \ell = d+1. \end{cases}$$

where d and λ_u are defined as in Lemma A.0.1.

We often denote the matrix $L_{G^\gamma} = L_G + \gamma I$ by K, and in particular use the notation L and K interchangeably.

Effective Resistance. Given a weighted graph $G = (V, E, w)$ we associate it with an electric circuit where the vertices are junctions and each edge e is a resistor of resistance $1/w(e)$. Now suppose in this circuit we inject one unit current at vertex u, extract one from vertex v, and let $\mathbf{f}_{uv} \in \mathbb{R}^m$ denote the currents induced on the edges. By Kirchhoff's current law, except for the source u and the sink v, the sum of the currents entering and exiting any vertex is zero. Hence, we have $\mathbf{b}_{uv} = B^\top \mathbf{f}_{uv}$. Let $\varphi \in \mathbb{R}^n$ denote the voltage potentials induced at the vertices in the above setting. By Ohm's law we have $\mathbf{f} = WB\varphi$. Putting these facts together:

$$\chi_u - \chi_v = B^\top WB\varphi = L\varphi.$$

Observe that $(\chi_u - \chi_v) \perp \ker(L)$, and hence $\varphi = L^+(\chi_u - \chi_v)$.

The *effective resistance* between vertices u and v in graph G, denoted by R_{uv} is defined as the voltage difference between vertices u and v, when a unit of current is injected into u and is extracted from v. Thus we have:

$$R_{uv} = \mathbf{b}_{uv}^\top L^+ \mathbf{b}_{uv}. \tag{1.1}$$

We also let $R_{uu} := 0$ for any $u \in V$, for convenience. For any matrix $K \in \mathbb{R}^{n \times n}$, we let $R_{uv}^K := \mathbf{b}_{uv}^\top K^+ \mathbf{b}_{uv}$.

Also, for any pair of vertices (w_1, w_2), the potential difference induced on this pair when sending a unit of flow from u to v can be calculated as:

$$\varphi(w_1) - \varphi(w_2) = \mathbf{b}_{w_1 w_2}^\top L^+ \mathbf{b}_{uv}. \tag{1.2}$$

Furthermore, if the graph is unweighted, the flow on edge (w_1, w_2) is

$$\mathbf{f}_{uv}(w_1 w_2) = \mathbf{b}_{w_1 w_2}^\top L^+ \mathbf{b}_{uv}. \tag{1.3}$$

We frequently use the following simple fact.

Fact 1.2.2 (See e.g. [7], Lemma 3)**.** *For any graph $G = (V, E, w)$, $\gamma \ge 0$ and any Laplacian matrix $L \in \mathbb{R}^V$, let $K = L + \gamma I$. Then, for any pair of vertices $(u, v), (u', v') \in V \times V$,*

$$|\mathbf{b}_{u'v'}^\top K^+ \mathbf{b}_{uv}| \le \mathbf{b}_{uv}^\top K^+ \mathbf{b}_{uv}.$$

Proof. Let $\varphi = K^+ \mathbf{b}_{uv}$. Suppose that for some $x \in V \setminus \{u\}$, $\varphi(x) > \varphi(u)$. Then, since $K = L + \gamma I$ is a full rank and diagonally dominant matrix, then one can easily see that we should have $b_{uv}(x) > 0$, which is a

contradiction. So, $\varphi(u) \geq \varphi(x)$ for any $x \in V \setminus \{u\}$. In a similar way, we can argue that $\varphi(v) \leq \varphi(y)$ for any $y \in V \setminus \{v\}$. So, the claim holds. □

Spectral Approximation. For matrices $C, D \in \mathbb{R}^{p \times p}$, we write $C \preceq D$, if $\forall x \in \mathbb{R}^p$, $x^\top Cx \leq x^\top Dx$. We say that $\widetilde{C}$ is $(1 \pm \epsilon)$-spectral sparsifier of C, and we write it as $\widetilde{C} \approx_\epsilon C$, if $(1-\epsilon)C \preceq \widetilde{C} \preceq (1+\epsilon)C$. Graph $\widetilde{G}$ is $(1 \pm \epsilon)$–spectral sparsifier of graph G if, $L_{\widetilde{G}} \approx_\epsilon L_G$. We also sometimes use a slightly weaker notation $(1-\epsilon)C \preceq_r \widetilde{C} \preceq_r (1+\epsilon)C$, to indicate that $(1-\epsilon)x^\top Cx \leq x^\top \widetilde{C}x \leq (1+\epsilon)x^\top Cx$, for any x in the row span of C.

1.3 Main result

We start by giving some intuition and presenting the high level idea of our algorithm in Section 1.3.1 below. In Section 1.3.2 we formally state the algorithm and provide correctness analysis. In Section 1.3.3 we describe how the required sketches can be implemented using the efficient pseudorandom number generator from [28]. Finally in Section 1.3.4 we give the proof of Theorem 1.1.1.

1.3.1 Overview of the approach

To illustrate our approach, suppose for now that our goal is to find edges with effective resistance at least $\frac{1}{\log n}$ in a graph $G = (V, E)$, which we denote by "heavy edges". This task has been studied in prior work on spectral sparsification [7] and was essentially shown in [28] to be sufficient to yield a spectral sparsification with only almost constant overhead. Each of [7] and [28] solve this problem by running ℓ_2-heavy hitters on approximate flow vectors, obtained by coarse sparsifier of the graph. The number of test flow vectors used in [7] is quadratic in the number of vertices, i.e., they brute force on all pair of vertices to find the heavy edges, and this was improved to $n^{1.4+o(1)}$ in [28]. Consequently, a natural question that one could attack to further improve the running times of these methods is the following:

Can we efficiently find a nearly linear number of test vectors that enable us to recover all heavy edges?

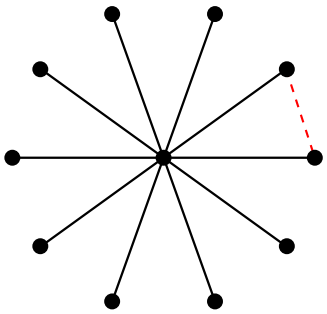
In this work, we answer this question in the affirmative and formally show that there exist a linear number of test vectors, which suffice to find all heavy edges. This is essentially the key technical contribution of this paper and generalizing this solution yields our main algorithmic results.

To illustrate our approach, suppose that one can compute the flow vector using the following formula [2]

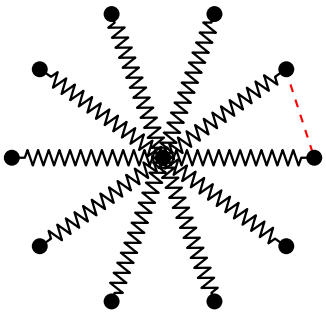
$$BL^+\mathbf{b}_{uv} = \mathbf{f}_{uv} \tag{1.4}$$

for any pair of vertices in polylogarithmic time (in our actual algorithms we will be unable to compute

[2]Note that in the actual algorithm we use K^+ as opposed to L^+, since we work with regularized versions of the Laplacian of G, denoted by K. We use L in this overview of our techniques to simplify notation.



(a) Graph of Example 1.3.1



(b) Graph of Example 1.3.2

Figure 1.1: (a) Graph of Example 1.3.1. A star with n petals along with one additional edge. (b) Graph of Example 1.3.2. A star graph with $\Theta(n^{0.7})$ petals, along with one additional edge. Each zigzag represents a path of connected cliques with effective resistance diameter $O(1)$.

these flow vectors exactly). Note that

$$||\mathbf{f}_{uv}||_2^2 = \mathbf{b}_{uv}^\top L^+ B^\top B L^+ \mathbf{b}_{uv} = \mathbf{b}_{uv}^\top L^+ \mathbf{b}_{uv} = R_{uv} \tag{1.5}$$

and

$$\mathbf{f}_{uv}(uv) = \mathbf{b}_{uv}^\top L^+ \mathbf{b}_{uv} = R_{uv}. \tag{1.6}$$

This implies that, when $R_{uv} > \frac{1}{\log n}$, the contribution of uv coordinate of this vector to the ℓ_2 norm is substantial, and known ℓ_2-heavy hitters can recover this edge using corresponding sketches, efficiently. One should note that ℓ_2-heavy hitter returns a set of edges with $\Omega(\frac{1}{\text{polylog}\, n})$ contribution to the ℓ_2^2 of the flow vector. A natural question that arises is whether it is possible to recover a heavy edge without using its flow vector, but rather using other flow vectors. Consider the following example.

Example 1.3.1 (Star Graph Plus Edge)**.** *Suppose that graph $G = (V, E)$ is a "star" with a center and n petals along with one additional edge that connects a pair of petals, i.e., $V = \{v_1, v_2, \ldots, v_n\}$ and $E = \{(v_1, v_2), (v_1, v_3), \cdot, (v_1, v_n)\} \cup \{(v_2, v_3)\}$ (see Figure 1.1a).*

Clearly, for edge (v_2, v_3), $R_{v_2 v_3} = \frac{2}{3}$. Suppose that we want to recover this edge by examining an electrical flow vector other than $\mathbf{f}_{v_2 v_3}$. We can in fact pick an arbitrary vertex $x \in V \setminus v_2$ and send one unit of flow to v_2. Regardless of the choice of x, edge (v_2, v_3) contributes an $\Omega(1)$ fraction of the energy of the flow, and thus can be recovered by applying heavy hitters to $\mathbf{f}_{x v_2}$. Similarly, for any v_i, when one unit of flow is sent from x to v_i, at least a constant fraction of the energy is contributed by edge (x, v_i). So, all high effective resistance edges in this graph (all edges) can be recovered using $n-1$ simple flow vectors, i.e., $\{\mathbf{f}_{x v_1}, \ldots, \mathbf{f}_{x v_n}\}$.

Of course, the graph in Example 1.3.1 has only n edges, and so could be stored explicitly in the streaming setting, without needing to recover edges from heavy hitter queries. However, we can give a similar example which is in fact dense.

Example 1.3.2 (Thick Star Plus Edge)**.** *Suppose that graph G is a dense version of the previous example as follows: it has a center and $\Theta(n^{0.7})$ petals. Each petal consists of a chain of $\Theta(n^{0.2})$ cliques of size $n^{0.1}$,*

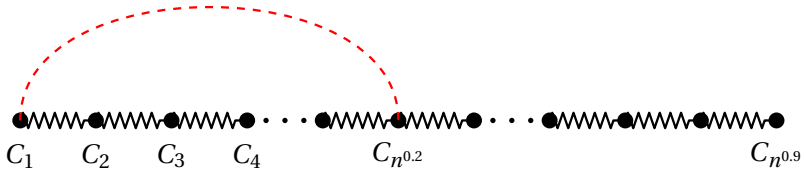


Figure 1.2: Graph of Example 1.3.3. Each C_i represents a cluster with $n^{0.1}$ vertices (with no internal edges) and each zigzag represents the edges of a complete bipartite graph between consecutive C_i's.

where each pair of consecutive cliques is connected with a complete bipartite graph. One can verify that the effective resistance diameter of each petal is $\Theta(1)$*. Now, we add an additional edge, e, that connects an arbitrary node in the leaf of one petal to a node in the leaf of another petal (see Figure 1.1b).*

As in Example 1.3.1, e is heavy, with $R_e = \Theta(1)$*. In fact, it is the only heavy edge in the graph. One can verify that, similar to Example 1.3.1, if we let* C_2 *and* C_3 *denote the cliques that e connects, choosing an arbitrary vertex x and sending flow to any node in* C_2 *and then to any node in* C_3*, will give an electrical flow vector where e contributes an* $\Omega(1)$ *fraction of the energy. Thus, e can be recovered by applying heavy hitters to these vectors. Consequently, using n test vectors (sending flow from x to each other node in the graph) one can recover all heavy edges of this example.*

Unfortunately, it is possible to give an example where the above simple procedure of checking the flow from an arbitrary vertex to all others fails.

Example 1.3.3 (Thick Line Plus Edge)**.** *Suppose that graph* $G = (V, E)$ *is a thick line, consisting of* $n^{0.9}$ *set of points (clusters) where any two consecutive clusters form a complete bipartite graph. Formally,* $V = \{v_1, v_2, \ldots, v_n\} = C_1 \cup C_2 \cup \cdots \cup C_{n^{0.9}}$*, where* C_i*'s are disjoint sets of size* $n^{0.1}$ *and*

$$E = \bigcup_{i=1}^{n^{0.9}-1} C_i \times C_{i+1}.$$

Also, add an edge $e = (u, v)$ *such that* $u \in C_1$ *and* $v \in C_{n^{0.2}}$ *(see Figure 1.2).*

One can verify that $R_e = \Omega(1)$*. However, if one picks an arbitrary vertex* $x \in V$ *and sends one unit of flow each other vertex, running* ℓ_2*-heavy hitters on each of these flows will not recover edge e if x is far from u and v in the thick path. Any flow that must cross* (u, v) *will have very large energy due to the fact that it must travel a long distance to the clusters containing these vertices, so e will not contribute non-trivial fraction.*

Fortunately, the failure of our recovery method in Example 1.3.3 is due to a simple fact: the effective resistance diameter of the graph is large. When the effective resistance diameter is small (as in Examples 1.3.1 and 1.3.2) the strategy always suffices. This follows from the following simple observation:

Observation 1.3.1. *For a graph* $G = (V, E)$*, suppose that for an edge* $e = (u, v) \in E$*, one has*

$$R_e \geq \beta.$$

Then, for any $x \in V$, in at least one of these settings, edge e carries at least $\beta/2$ units of flow:

1. *One unit of flow is sent from x to u.*
2. *One unit of flow is sent from x to v.*

This observation follows formally from the following simple lemma.

Lemma 1.3.1. *For a graph $G = (V, E)$, suppose that $D \in \mathbb{R}^{V \times V}$ is a PSD matrix. Then, for any pair of vertices $(u, v) \in \binom{V}{2}$ and for any vertex $x \in V \setminus \{u, v\}$,*

$$\max\{|\mathbf{b}_{xu}^\top D \mathbf{b}_{uv}|, |\mathbf{b}_{xv}^\top D \mathbf{b}_{uv}|\} \geq \frac{\mathbf{b}_{uv}^\top D \mathbf{b}_{uv}}{2}.$$

Proof. Note that

$$\mathbf{b}_{uv}^\top D \mathbf{b}_{uv} = \left(\mathbf{b}_{ux}^\top + \mathbf{b}_{xv}^\top\right) D \mathbf{b}_{uv} = \mathbf{b}_{ux}^\top D \mathbf{b}_{uv} + \mathbf{b}_{xv}^\top D \mathbf{b}_{uv},$$

and hence, the claim holds. □

Consider the setting where $\beta = \frac{1}{\log n}$. The observation guarantees that edge e contributes at least $\frac{1}{4\log^2 n}$ energy to either flow $\mathbf{f}_{xv}$ or $\mathbf{f}_{xu}$. Thus, we can recover this edge via ℓ_2 heavy hitters, as long as the total energy $\|\mathbf{f}_{xv}\|_2^2$ or $\|\mathbf{f}_{xu}\|_2^2$ is not too large. Note that this energy is just equal to the effective resistance R_{xv} between x and v (respectively x and u). Thus it is bounded if the effective resistance diameter is small, demonstrating that our simple recovery procedure always succeeds in this setting. For example, if the diameter is $O(1)$, both $\|\mathbf{f}_{xv}\|_2^2 = O(1)$ and $\|\mathbf{f}_{xu}\|_2^2 = O(1)$, and so by Observation 1.3.1, edge $e = (u, v)$ contributes at least a $\Theta\left(\frac{1}{\log^2 n}\right)$ fraction of the energy of at least one these flows.

We next explain how to extend this procedure to handle general graphs, like that of Example 1.3.3.

Ball carving in effective resistance metric: When the effective resistance diameter of G is large, if we attempt to recover e using ℓ_2-heavy hitters on the flow vectors $\mathbf{f}_{xu}$ and $\mathbf{f}_{xv}$, for an arbitrary chosen $x \in V$, we may fail if the effective resistance distance between x and v or u ($\|\mathbf{f}_{xv}\|_2^2$ or $\|\mathbf{f}_{xu}\|_2^2$) is large. This is exactly what we saw in Example 1.3.3.

However, using the fact that $||\mathbf{f}_{xv}||_2^2 = R_{xv}$, our test will succeed if we find a vertex x, which is close to u and v in the effective resistance metric. This suggests that we should partition the vertices into cells of fairly small effective resistance diameter, ensuring that both endpoints of an edge (u, v) that we would like to recover fall in the same cell with nontrivially large probability. This is exactly what standard metric decomposition techniques achieve through a ball-carving approach, which we use, as described next.

Partitioning the graph into low effective resistance diameter sets: It is well-known that using Johnson-Lindenstrauss (JL) dimension reduction (see Lemma A.0.3), one can embed vertices of a graph in $\mathbb{R}^q$, for $q = O(\log n)$, such that the Euclidean distance squares correspond to a constant factor multiplicative

approximation to effective resistance of corresponding vertices. We then partition $\mathbb{R}^q$ intro ℓ_∞ balls centered at points of a randomly shifted infinite q-dimensional grid with side length $w > 0$, essentially defining a hash function that maps every point in $\mathbb{R}^q$ to the nearest point on the randomly shifted grid. We then bound the maximum effective resistance of pair of vertices in the same bucket (see Claim 1.3.2), and show how an appropriate choice of the width w ensures that u and v belong to the same cell, with a probability no less than a universal constant (see Claim 1.3.1). This ensures that in at least one of $O(\log n)$ independent repetitions of this process with high probability, u and v fall into the same cell. We note that the parameters of our partitioning scheme can be improved somewhat using Locality Sensitive Hashing techniques (e.g., [58, 59, 60, 61, 62]). More precisely, LSH techniques would improve the space complexity by polylogarithmic factors at the expense of slightly higher runtime (the best improvement in space complexity would result from Euclidean LSH [60, 62], at the cost of an $n^{o(1)}$ additional factor in runtime). However, since the resulting space complexity does not quite match the lower bound of $\Omega(n\log^3 n)$ due to [15], we leave the problem of fine-tuning the parameters of the space partitioning scheme as an exciting direction for further work.

Sampling edges with probability proportional to effective resistances: The above techniques can actually be extended to recover edges of any specific target effective resistance. Broadly speaking, if we aim to capture edges of effective resistance about R, we can afford to lower our grid cell size proportionally to R. Unfortunately, these edges don't contribute enough to the flow vector to be recoverable. Thus, we will also subsample the edges of the graph at rate approximately proportional to R to allow us to detect the target edges while also subsampling them.

1.3.2 Our algorithm and proof of main result

As mentioned in the introduction, our algorithm consists of two phases. In the first phase, our algorithm maintains sketches of the stream, updating the sketches at each edge addition or deletion. Then, in the second phase, when queried, it can recover a spectral sparsifier of the graph from the sketches that have been maintained in the first phase. In the following lines, we give a brief overview of each phase:

Updating sketches in the dynamic stream. Our algorithm maintains a set of sketches ΠB, of size $O(n\,\text{polylog}(n)\cdot\epsilon^{-2})$, and updates them each time it receives an edge addition or deletion in the stream. ΠB consists of multiple sketches $(\Pi_s^\ell B_s^\ell)_{\ell,s}$ where B_s^ℓ is a subsampling of the edges in B at rate 2^{-s} and Π_s^ℓ is an ℓ_2 heavy hitters sketch. In section 1.3.3 we discuss these sketches in more detail and we show that the update time for each edge addition or deletion is $O(\text{polylog}(n)\cdot\epsilon^{-2})$.

Recursive sparsification: After receiving the updates in the form of a dynamic stream, as described above, our algorithm uses the maintained sketches to recover a spectral sparsifier of the graph. This is done recursively, and heavily relies on the idea of a chain of coarse sparsifiers described in Lemma A.0.1. For a regularization parameter ℓ between 0 and $d = O(\log n)$ the task of $\textsc{Sparsify}(\Pi^{\leq\ell}B, \ell, \epsilon)$ is to output a

spectral sparsifier to matrix L_ℓ, which is defined as follows:

$$L_\ell = \begin{cases} L_G + \frac{\lambda_u}{2^\ell} I & \text{if } 0 \le \ell \le d \\ L_G & \text{if } \ell = d+1. \end{cases}$$

where $d = \lceil \log_2 \frac{\lambda_u}{\lambda_\ell} \rceil$ (see Lemma A.0.1 for more details about chain of coarse sparsifiers). Note that the call receives a collection of sketches $\Pi^{\le \ell} B$ as input that suffices for all recursive calls with smaller values of ℓ. So, in order to get a sparsifier of the graph we invoke SPARSIFY($\Pi^{\le d+1} B, d+1, \epsilon$), which receives all the sketches maintained throughout the stream and passes the required sketches to the recursive calls in line 6 of Algorithm 1. This recursive algorithm takes as input $\Pi^{\le \ell} B$ corresponding to the parts of the sketch used to recover a spectral approximation to L_k for all $k \le \ell$, ℓ corresponding to the current L_ℓ which we wish to recover a sparsifier of, and ϵ corresponding to the desired sparsification accuracy. The algorithm first invokes itself recursively to recover $\tilde{K}$, a spectral approximation for $L_{\ell-1}$ (or uses the trivial approximation $\lambda_u I$ when $\ell = 0$). The effective resistance metric induced by $\tilde{K}$ is then approximated using the Johnson-Lindenstrauss lemma (JL). Finally, the procedure RECOVEREDGES (i.e. Algorithm 2) uses this metric and the heavy hitters sketches $(\Pi_s^\ell B_s^\ell)_s$. We formally state our algorithm, Algorithm 2 below.

Algorithm 1 SPARSIFY($\Pi^{\le \ell} B, \ell, \epsilon$)

1: **procedure** SPARSIFY($\Pi^{\le \ell} B, \ell, \epsilon$)
2: $W \leftarrow 0^{n \times n}$
3: **if** $\ell = 0$ **then**
4: $\widetilde{K} \leftarrow \lambda_u I$
5: **else**
6: $\widetilde{K} \leftarrow \frac{1}{2(1+\epsilon)}$SPARSIFY($\Pi^{\le \ell-1} B, \ell-1, \epsilon$)
7: $\widetilde{B} \leftarrow$ the edge vertex incident matrix of $\widetilde{K}$ (discarding the regularization)
8: $\widetilde{W} \leftarrow$ the diagonal matrix of weights over the edges of $\widetilde{K}$ (discarding the regularization)
9: $Q \leftarrow q \times \binom{n}{2}$ is a random ± 1 matrix for $q \leftarrow 1000 \log n$
10: ▷ *q above is chosen as it suffices to get a $(1 \pm \frac{1}{5})$ approximation from JL*
11: $M \leftarrow \frac{1}{\sqrt{q}} Q \widetilde{W}^{1/2} \widetilde{B} \widetilde{K}^+$ ▷ *M is such that $R^{\widetilde{K}}_{uv} \le \frac{5}{4} ||M(\chi_u - \chi_v)||_2^2 \le \frac{3}{2} R^{\widetilde{K}}_{uv}$ w.h.p.*
12: ▷ *$R^{\widetilde{K}}$ is the effective resistance metric in $\widetilde{K}$*
13: **for** $s \in [-\log(3 \cdot c_2 \cdot \log n \cdot \epsilon^{-2}), 10 \log n]$ **do**
14: $E_s \leftarrow$ RECOVEREDGES($\Pi^\ell_{s^+} B^\ell_{s^+}, M, \widetilde{K}^+, s, q, \epsilon$) ▷ *We use the notation $s^+ = max(0, s)$*
15: **for** $e \in E_s$ **do**
16: $W(e,e) \leftarrow 2^{+(s^+)}$
17: **if** $\ell = \left\lceil \log_2 \frac{\lambda_u}{\lambda_\ell} \right\rceil + 1$ **then**
18: $\gamma \leftarrow 0$
19: **else**
20: $\gamma \leftarrow \frac{\lambda_u}{2^\ell}$
21: **return** $B_n^\top W B_n + \gamma I$.

Algorithm 2 (the RECOVEREDGES primitive) is the core of Algorithm 1. It receives a parameter s as input,

and its task is to recover edge of effective resistance $\approx \frac{\epsilon^2}{\log n} 2^{-s}$ from a sample at rate $\min(1, O(\frac{1}{\epsilon^2}\log n \cdot 2^{-s}))$ from an appropriate sketch. It is convenient to let s range from $-O(\log(\log n/\epsilon^2))$ to $O(\log n)$, so that the smallest value of s corresponds to edges of constant effective resistance. That way the sampling level corresponding to s is simply equal to $s^+ := \max(0, s)$. Therefore Algorithm 2 takes as input a heavy hitters sketch $\Pi^\ell_{s^+} B^\ell_{s^+}$ of $B^\ell_{s^+}$, the edge incidence matrix of L_ℓ sampled at rate 2^{-s^+}, an approximate effective resistance embedding M, the target sampling probability 2^{-s}, the dimension q of the embedding, and the target accuracy ϵ. This procedure then performs the previously described random grid hashing of the points using the effective resistance embedding and queries the heavy hitters sketch to find the edges sampled at the appropriate rate.

The development and analysis of RECOVEREDGES (Algorithm 2) is the main technical contribution of our paper. In the rest of the section we prove correctness of Algorithm 2 (Lemma 1.3.2, our main technical lemma), and then provide a correctness proof for Algorithm 1, establishing Theorem 1.3.4. We then put these results together with runtime and space complexity bounds to obtain a proof of Theorem 1.1.1.

Algorithm 2 RECOVEREDGES($\Pi^\ell_{s^+} B^\ell_{s^+}, M, \widetilde{K}^+, s, q, \epsilon$)

1: **procedure** RECOVEREDGES($\Pi^\ell_{s^+} B^\ell_{s^+}, M, \widetilde{K}^+, s, q, \epsilon$)
2: $E' \leftarrow \emptyset$. ▷ *q is the dimension to perform hashing in, s is the sampling level*
3: $C \leftarrow$ the constant in the proof of Lemma 1.3.2
4: $c_2 \leftarrow$ the oversampling constant of Theorem A.0.1
5: $w \leftarrow 2q \cdot \sqrt{\frac{\epsilon^2}{c_2 \cdot 2^s \cdot \log n}}$.
6: **for** $j \in [10 \log n]$ **do**
7: For each dimension $i \in [q]$, choose $s_i \sim \text{Unif}([0, w])$.
8: Initialize $H \leftarrow \emptyset$ to an empty hash table
9: **for** $u \in V$ **do** ▷ *Hash vertices to points on randomly shifted grid*
10: For all $i \in [q]$, let $\mathcal{G}(u)_i := \left\lfloor \frac{(M\chi_u)_i - s_i}{w} \right\rfloor$.
11: Insert u into H with key $\mathcal{G}(u) \in \mathbb{Z}^q$
12: ▷ *$\mathcal{G}(u) \in \mathbb{Z}^q$ indexes a point on a randomly shifted grid*
13: **for** $b \in \text{keys}(H)$ **do** ▷ *$b \in \mathbb{Z}^q$ indexes a point on a randomly shifted grid*
14: $x \leftarrow$arbitrary vertex in $H^{-1}(b)$
15: **for** $v \in H^{-1}(b) \setminus \{x\}$ **do**
16: $F \leftarrow \text{HEAVYHITTER}\left(\Pi^\ell_{s^+} B^\ell_{s^+} \widetilde{K}^+ \mathbf{b}_{xv}, \frac{1}{2} \cdot \frac{1}{C \cdot q^3} \cdot \sqrt{\frac{\epsilon^2}{\log n}}\right)$. ▷ *As per Lemma A.0.2*
17: **for** $e \in F$ **do**
18: $p'_e \leftarrow \frac{5}{4} \cdot c_2 \cdot ||Mb_e||_2^2 \cdot \log n/\epsilon^2$
19: **if** $p'_e \in (2^{-s-1}, 2^{-s}]$ **then**
20: $E' \leftarrow E' \cup \{e\}$.
21: **return** E'.

Lemma 1.3.2 below is our main technical lemma. Specifically, Lemma 1.3.2 proves that if Algorithm 1 successfully executes all lines before line 13, then each edge is sampled and weighted properly (as required by Theorem A.0.1), in the remaining steps.

Lemma 1.3.2 (Edge Recovery)**.** *Consider an invocation of* RECOVEREDGES($\Pi^\ell_{s^+} B^\ell_{s^+}, M, \widetilde{K}, s, q, \epsilon$) *of Algorithm 2, where $\Pi^\ell_{s^+} B^\ell_{s^+}$ is a sketch of the edge incidence matrix B of the input graph G as described in Section 1.3.3, s is some integer, and $\epsilon \in (0, 1/5)$. Suppose further that $\widetilde{K}$ and M satisfy the following guaran-*

tees:

(A) *$\widetilde{K}$ is such that $\frac{1}{3}\cdot L_\ell \preceq_r \widetilde{K} \preceq_r L_\ell$ (see lines 4 and 6 of Algorithm 1)*

(B) *M is such that for any pair of vertices u and v, $R^{\widetilde{K}}_{uv} \le \frac{5}{4}||M(\chi_u-\chi_v)||_2^2 \le \frac{3}{2}R^{\widetilde{K}}_{uv}$ ($R^{\widetilde{K}}$ is the effective resistance metric in $\widetilde{K}$; see line 11 of Algorithm 1)*

Then, with high probability, for every edge e, RECOVEREDGES$(\Pi^{\ell}_{s^+}B^{\ell}_{s^+}, M, \widetilde{K}, s, q, \epsilon)$ *will recover e if and only if:*

(1) *$\frac{5}{4}\cdot c_2\cdot||Mb_e||_2^2\cdot\log(n)/\epsilon^2 \in (2^{-s-1},2^{-s}]$ where c_2 is the oversampling constant of Theorem A.0.1 (see lines 18 and 19 of Algorithm 2), and*

(2) *edge e is sampled in $B^{\ell}_{s^+}$.*

The proof of Lemma 1.3.2 relies on the following two claims regarding the hashing scheme of Algorithm 2. First, Claim 1.3.1 shows that the endpoints of an edge of effective resistance bounded by a threshold most likely get mapped to the same grid point in the random hashing step in line 10 of Algorithm 2.

Claim 1.3.1 (Hash Collision Probability)**.** *Let q be a positive integer and let the function $\mathcal{G}:\mathbb{R}^q\to\mathbb{Z}^q$ define a hashing with width $w>0$ as follows:*

$$\forall\, i\in[q],\ \ \mathcal{G}(u)_i=\left\lfloor\frac{u_i-s_i}{w}\right\rfloor$$

where $s_i\sim$ Unif$[0,w]$, as per line 10 of Algorithm 2. If for a pair of points $x,y\in\mathbb{R}^q$, $||x-y||_2\le w_0$ and $w\ge 2w_0q$, then $\mathcal{G}(x)=\mathcal{G}(y)$ with probability at least $1/2$.

Proof. First note that by union bound

$$\mathbb{P}(\mathcal{G}(x)\neq\mathcal{G}(y))=\mathbb{P}(\exists i:\mathcal{G}(x)_i\neq\mathcal{G}(y)_i)\le\sum_{i=1}^{q}\mathbb{P}(G(x)_i\neq G(y)_i)\,. \tag{1.7}$$

Now let us bound each term of the sum.

$$\begin{aligned}\mathbb{P}(\mathcal{G}(x)_i\neq\mathcal{G}(y)_i)&=\mathbb{P}\left(\left\lfloor\frac{x_i-s_i}{w}\right\rfloor\neq\left\lfloor\frac{y_i-s_i}{w}\right\rfloor\right)\\&=\frac{|x_i-y_i|}{w}\\&\le\frac{||x-y||_2}{w}\\&\le\frac{1}{2q}\end{aligned} \tag{1.8}$$

Combining (1.7) and (1.8), we get that $\mathbb{P}(\mathcal{G}(x)\neq\mathcal{G}(y))\le 1/2$ as claimed. □

The next claim, Claim 1.3.2 bounds the effective resistance diameter of buckets in the hash table constructed in line 11 of Algorithm 2.

Claim 1.3.2 (Hash Bucket Diameter)**.** *Let the function $\mathcal{G}:\mathbb{R}^q \to \mathbb{Z}^q$, for some integer q, define a hashing with width $w>0$ as follows:*

$$\forall i \in [q],\ \ \mathcal{G}(u)_i = \left\lfloor \frac{u_i - s_i}{w} \right\rfloor$$

where $s_i \sim \text{Unif}[0,w]$, as per line 10 of Algorithm 2. For any pair of points $u,v \in \mathbb{R}^q$, such that $\mathcal{G}(u)=\mathcal{G}(v)$, one has

$$||u-v||_2 \le w\cdot\sqrt{q}.$$

Proof. Since $\mathcal{G}(u)=\mathcal{G}(v)$, then

$$\begin{aligned} ||u-v||_2 &= \sqrt{\sum_{i\in q}(u_i-v_i)^2} \\ &\le \sqrt{w^2\cdot q} && \text{since } \forall i\in q, |u_i-v_i|\le w \\ &= w\cdot\sqrt{q}. \end{aligned}$$

□

Using Claim 1.3.1 and Claim 1.3.2 we now prove Lemma 1.3.2.

Proof of Lemma 1.3.2: Let $p'_e := \frac{5}{4}\cdot c_2\cdot||Mb_e||_2^2\cdot\log(n)/\epsilon^2$. First note that both of conditions **(1)** and **(2)** are necessary. Indeed, if e is not sampled in B_s, it will never be returned by HeavyHitter in line 16, and if $p'_e \notin (2^{-s-1},2^{-s}]$ then e will not be added to E' due to line 19 of Algorithm 2. It remains to show that the two conditions are sufficient to recover e with high probability.

For an edge $(u,v)=e\in E$ satisfying conditions **(1)** and **(2)** we prove that the size of the grid (w as defined in line 5 of Algorithm 2) is large enough to capture edge e, as described by Claim 1.3.1. Specifically, we invoke the claim with $w_0 = ||Mb_e||_2$. Note that we have $w\ge 2qw_0$ by the setting of w in line 5 and the fact that

$$||Mb_e||_2 = \sqrt{\frac{4}{5}\cdot p'_e\cdot\frac{\epsilon^2}{c_2\cdot\log n}} \le \sqrt{\frac{\epsilon^2}{c_2\cdot 2^s\cdot\log n}},$$

where we used the fact that $p'_e\le 2^{-s}$. Thus, we have $w\ge 2qw_0$ as prescribed by Claim 1.3.1, so u and v fall into the same cell with probability at least $1/2$ in a single instance of hashing. Hashing is then repeated $10\log n$ times to guarantee that they fall into the same cell at least once with high probability, see line 6 of Algorithm 2.

Consider now an instance of hashing where u and v fall into the same cell, say $\mathscr{C}$ (which corresponds to a hash bucket in our hash table H). Let x be chosen arbitrarily from $\mathscr{C}$ as per line 14 of Algorithm 2 . Our algorithm sends electrical flow from x to both u and v and by Observation 1.3.1 in at least one of these flows e will have weight $R_e^{\widetilde{K}}/2$. More precisely, by Lemma 1.3.1 invoked with $D=\widetilde{K}^+$ we have

$$\max\{|\mathbf{b}_{xu}^\top\widetilde{K}^+\mathbf{b}_{uv}|,|\mathbf{b}_{xv}^\top\widetilde{K}^+\mathbf{b}_{uv}|\} \ge \frac{\mathbf{b}_{uv}^\top\widetilde{K}^+\mathbf{b}_{uv}}{2} = R_e^{\widetilde{K}}/2. \tag{1.9}$$

Without loss of generality assume that this is the flow from x to v.

It remains to show, that unlike in Example 1.3.3, the total energy of the xv flow does not overshadow the contribution of edge e. Intuitively this is because the effective resistence of e is proportional to $2^{-s}\cdot\epsilon^2$ and therefore its ℓ_2-contribution is proportional to $2^{-2s}\cdot\epsilon^4$. On the other hand, the effective resistence diameter of $\mathscr{C}$ is proportional to $2^{-s}\cdot\epsilon^2$, which bounds the energy of the xv flow before subsampling. Subsampling at rate 2^{-s} decreases the energy by a factor of 2^{-s} in expectation, and the energy concentrates sufficiently around its expectation with high probability. We prove everything in more detail below. It turns out that the actual ratio between contribution of e and the entire energy of the subsampled flow is polylogarithmic in n and quadratic in ϵ. Therefore, we can afford to store a heavy hitter sketch powerful enough to recover e.

Now let $\widetilde{\mathbf{f}}_{xv} = B\widetilde{K}^+(\chi_x-\chi_v)$, and $\widetilde{\mathbf{f}}_{xu} = B\widetilde{K}^+(\chi_x-\chi_u)$. Note that $\mathbf{f}_{xv}\in\mathbb{R}^{\binom{n}{2}}$ is a vector whose nonzero entries are exactly the voltage differences across edge in G when one unit of current is forced from x to v in $\widetilde{K}$. We have, writing L instead of L_ℓ to simplify notation,

$$\begin{aligned}
||\widetilde{\mathbf{f}}_{xv}||_2^2 &= (\chi_x-\chi_v)^\top\widetilde{K}^+B^\top B\widetilde{K}^+(\chi_x-\chi_v)\\
&\le (\chi_x-\chi_v)^\top\widetilde{K}^+L\widetilde{K}^+(\chi_x-\chi_v) && \text{Since } B^\top B\preceq L\\
&\le 3\cdot(\chi_x-\chi_v)^\top\widetilde{K}^+\widetilde{K}\widetilde{K}^+(\chi_x-\chi_v) && \text{Since } L\preceq 3\cdot\widetilde{K} \text{ by assumption } \textbf{(A)}\\
&= 3\cdot(\chi_x-\chi_v)^\top\widetilde{K}^+(\chi_x-\chi_v) && \text{of the lemma}\\
&= 3\cdot R_{xv}^{\widetilde{K}}
\end{aligned}$$

Moreover we have

$$\widetilde{\mathbf{f}}_{xv}(uv) = (\chi_u-\chi_v)^\top\widetilde{K}^+(\chi_x-\chi_v)$$

and

$$\widetilde{\mathbf{f}}_{xu}(uv) = (\chi_u-\chi_v)^\top\widetilde{K}^+(\chi_x-\chi_u).$$

For simplicity, let

$$\beta := \frac{\epsilon^2}{c_2\cdot\log n}.$$

By (1.9) we have

$$\begin{aligned}
|\widetilde{\mathbf{f}}_{xv}(uv)| &\ge \frac{b_{uv}^\top\widetilde{K}^+b_{uv}}{2}\\
&\ge \frac{5}{12}\cdot||M\mathbf{b}_e||_2^2 && \text{By assumption } \textbf{(B)} \text{ of the lemma}\\
&\ge \frac{1}{3}\cdot\frac{1}{2^{s+1}\cdot c_2}\cdot\frac{\epsilon^2}{\log n} && \text{Since } p'_e\ge\frac{1}{2^{s+1}}\\
&= \frac{1}{3}\cdot\frac{\beta}{2^{s+1}}
\end{aligned} \tag{1.10}$$

Since x, v belong to the same cell, by Claim 1.3.2, $||M(\chi_u - \chi_v)||_2 \le w \cdot \sqrt{q}$, thus,

$$\begin{aligned}||\widetilde{\mathbf{f}}_{xv}||_2^2 &\le 3 \cdot R_{xv}^{\widetilde{K}} \\ &\le \frac{5}{4}(w^2 \cdot q) \qquad \text{Since } R_{xv}^{\widetilde{K}} \le \frac{15}{4}||M(\chi_x - \chi_v)||_2^2 \text{ by } \textbf{(B)} \\ &= 15q^3 \cdot \frac{\epsilon^2}{c_2 \cdot 2^s \cdot \log n} \quad \text{By line 5 of Algorithm 2} \\ &= 15q^3 \cdot \frac{\beta}{2^s}\end{aligned} \tag{1.11}$$

Now, let $\widetilde{\mathbf{f}}_{xv}^{(s)} := B_s \widetilde{K} \mathbf{b}_{xv}$ denote an independent sample of the entries of $\widetilde{\mathbf{f}}_{xv}$ with probability $\frac{1}{2^s}$. We now argue that, if the edge (u, v) is included in B_s, then it is recovered with high probability by the heavy hitter procedure HEAVYHITTER in line 16. We let $\widetilde{\mathbf{f}}^{(s)} := \widetilde{\mathbf{f}}_{xv}^{(s)}$ and $\widetilde{\mathbf{f}} := \widetilde{\mathbf{f}}_{xv}$ (i.e., we omit the subscript xv) to simplify notation.

We will prove a lower bound on $\frac{\widetilde{\mathbf{f}}^{(s)}(uv)^2}{||\widetilde{\mathbf{f}}^{(s)}||_2^2}$ that holds with high probability. Note that

$$||\widetilde{\mathbf{f}}^{(s)}||_2^2 = \sum_{e \in B_s \setminus \{(u,v)\}} \widetilde{\mathbf{f}}^{(s)}(e)^2 + \widetilde{\mathbf{f}}^{(s)}(uv)^2 \tag{1.12}$$

For ease of notation let $X := \sum_{e \in B_s \setminus \{(u,v)\}} \widetilde{\mathbf{f}}^{(s)}(e)^2$, and let $\tau := R_{xv}^{\widetilde{K}}$. Thus, we have for a sufficiently large constant $C > 1$

$$\begin{aligned}&\Pr\left(\frac{\widetilde{\mathbf{f}}^{(s)}(uv)^2}{||\widetilde{\mathbf{f}}^{(s)}||_2^2} < \frac{1}{C^2 \cdot q^6} \cdot \frac{\epsilon^2}{\log n} \middle| (u,v) \in B_s\right) \\ =&\Pr\left(X > \left(\frac{C^2 \cdot q^6 \cdot \log n}{\epsilon^2} - 1\right) \cdot \widetilde{\mathbf{f}}^{(s)}(uv)^2 \middle| (u,v) \in B_s\right) \\ \le&\Pr\left(||\widetilde{\mathbf{f}}^{(s)}||_2^2 > \frac{1}{2} \cdot C^2 \cdot q^6 \cdot \frac{\log n}{\epsilon^2} \cdot \widetilde{\mathbf{f}}(uv)^2\right) \\ \le&\Pr\left(\left|\left|\frac{\widetilde{\mathbf{f}}^{(s)}}{\tau}\right|\right|_2^2 > \frac{1}{\tau^2} \cdot \frac{1}{2} \cdot C^2 \cdot q^6 \cdot \frac{\log n}{\epsilon^2} \cdot \widetilde{\mathbf{f}}(uv)^2\right) \\ =&\Pr\left(||\widetilde{\mathbf{y}}^{(s)}||_2^2 > \frac{1}{\tau^2} \cdot \frac{1}{2} \cdot C^2 \cdot q^6 \cdot \frac{\log n}{\epsilon^2} \cdot \widetilde{\mathbf{f}}(uv)^2\right),\end{aligned} \tag{1.13}$$

where we let $\widetilde{\mathbf{y}} := \frac{\widetilde{\mathbf{f}}}{\tau}$ and $\widetilde{\mathbf{y}}^{(s)} := \frac{\widetilde{\mathbf{f}}^{(s)}}{\tau}$ to simplify notation in the last line and used the fact that $\widetilde{\mathbf{f}}^{(s)}(uv)^2 = \widetilde{\mathbf{f}}(uv)^2$ conditioned on $(u, v) \in B_s$ in going from line 2 to line 3. Noting that $|\widetilde{\mathbf{f}}(uv)| \ge \frac{1}{3} \cdot \frac{\beta}{2^{s+1}}$ by (1.10) and $\tau \le 5q^3 \cdot \frac{\beta}{2^s}$ by (1.11), we get that the last line in (1.13) is upper bounded by

$$\Pr\left(\frac{\widetilde{\mathbf{f}}^{(s)}(uv)^2}{||\widetilde{\mathbf{f}}^{(s)}||_2^2} < \frac{1}{C^2 \cdot q^6} \cdot \frac{\epsilon^2}{\log n} \middle| (u,v) \in B_s\right) \le \Pr\left(||\widetilde{\mathbf{y}}^{(s)}||_2^2 > \frac{C' \cdot \log n}{\epsilon^2}\right), \tag{1.14}$$

where C' is a constant that can be made arbitrarily large by increasing C. On the other hand, we have the

following

$$\begin{aligned}
\mathbf{E}\left(||\widetilde{\mathbf{y}}^{(s)}||_2^2\right) &= \frac{1}{2^s}\cdot\frac{||\widetilde{\mathbf{f}}||_2^2}{\tau^2} && \text{Since } \widetilde{\mathbf{f}}^{(s)} \text{ is obtained by sampling at rate } \frac{1}{2^s}\\
&\le \frac{1}{2^s}\cdot\frac{3}{\tau} && \text{By (1.11)}\\
&\le \frac{1}{2^s}\cdot\frac{6}{R^{\widetilde{K}}_{uv}} && \text{By (1.9) and Fact 1.2.2}\\
&\le \frac{1}{2^s}\cdot\frac{36}{5\cdot||M\mathbf{b}_{uv}||_2^2} && \text{By assumption } \textbf{(B)} \text{ of the lemma}\\
&\le \frac{1}{2^s}\cdot\frac{36}{5\cdot\frac{4}{5\cdot c_2}\cdot\frac{1}{2^{s+1}}\cdot\frac{\epsilon^2}{\log n}} && \text{By condition } \textbf{(1)} \text{ of the lemma}\\
&= \frac{18\cdot c_2\cdot\log n}{\epsilon^2},
\end{aligned}$$

where the transition from line 2 to line 3 is justified by noting that

$$\tau \ge \left|(\chi_u-\chi_v)^\top \widetilde{K}^+(\chi_x-\chi_v)\right| \ge \frac{1}{2}(\chi_u-\chi_v)^\top \widetilde{K}^+(\chi_u-\chi_v) = \frac{1}{2}R^{\widetilde{K}}_{uv}$$

by Fact 1.2.2 and choice of x.

We now upper bound the right hand side of (1.14). For every entry (a,b) in $\widetilde{\mathbf{y}}$, using Fact 1.2.2 one has

$$\left|\widetilde{\mathbf{y}}_{ab}\right| = \frac{|(\chi_a-\chi_b)^\top \widetilde{K}^+(\chi_x-\chi_v)|}{\tau} \le \frac{|(\chi_x-\chi_v)^\top \widetilde{K}^+(\chi_x-\chi_v)|}{\tau} = 1$$

Thus, every entry is in $[-1,1]$, and since every entry is sampled independently, so we can use standard Chernoff/Hoeffding [63] bound and we get

$$\Pr\left(\left|\left|\widetilde{\mathbf{y}}^{(s)}\right|\right|_2^2 > \frac{C'\cdot\log n}{\epsilon^2}\right) \le n^{-10}$$

as long as C' is a sufficiently large absolute constant (which can be achieved by making the constant C sufficiently large). Hence, we get from (1.13) that with high probability over the sampling of entries in B_s

$$\frac{|\widetilde{\mathbf{f}}^{(s)}(uv)|}{||\widetilde{\mathbf{f}}^{(s)}||_2} \ge \frac{1}{C\cdot q^3}\cdot\sqrt{\frac{\epsilon^2}{\log n}}.$$

We set $\eta = \frac{1}{2}\cdot\frac{1}{C\cdot q^3}\cdot\sqrt{\frac{\epsilon^2}{\log n}}$, thus if $|\widetilde{\mathbf{f}}^{(s)}(uv)| \ge 2\eta||\widetilde{\mathbf{f}}^{(s)}||_2$ our sparse recovery sketch must return uv with high probability, by Lemma A.0.2. □

Theorem 1.3.4. *(Correctness of Algorithm 1) Algorithm* SPARSIFY$(\Pi^{\le\ell}B,\ell,\epsilon)$*, for* $\ell = d+1 = \lceil\log_2\frac{\lambda_u}{\lambda_\ell}\rceil+1$ *(see Lemma A.0.1), any* $\epsilon\in(0,1/5)$ *and sketches* $\Pi^{\le\ell}B$ *of graph* G *as described in Section 1.3.3, returns a graph* H *with* $O(n\cdot\text{polylog}\,n\cdot\epsilon^{-2})$ *weighted edges, with Laplacian matrix* L_H*, such that*

$$L_H \approx_\epsilon L_G,$$

with high probability.

Proof. Let $\gamma = \lambda_u/2^\ell$. As the algorithm only makes recursive calls with lower values of ℓ, we proceed by induction on ℓ.

Inductive hypothesis: A call of SPARSIFY($\Pi^{\leq \ell} B, \ell, \epsilon$) returns a graph H^ℓ with $O(n \cdot \text{polylog}\, n \cdot \epsilon^{-2})$ weighted edges, with Laplacian matrix L_{H^ℓ}, such that

$$L_{H^\ell} \approx_\epsilon L_\ell$$

with high probability, where

$$L_\ell = \begin{cases} L_G + \frac{\lambda_u}{2^\ell} I & \text{if } 0 \leq \ell \leq d \\ L_G & \text{if } \ell = d+1. \end{cases}$$

and for all $\ell \geq 0$ the matrix $\widetilde{K}$ defined at the beginning of Algorithm 1 is a 3-spectral sparsifier of $G^{\gamma(\ell)}$.

Base case: $\ell = 0$. In this case we set $\widetilde{K} = \lambda_u I$ (see line 4 of Algorithm 1). By Lemma A.0.1 we have

$$\frac{1}{2} \cdot L_\ell \preceq \widetilde{K} \preceq L_\ell, \tag{1.15}$$

i.e. $\widetilde{K}$ is a factor 3 spectral approximation of L_ℓ for $\ell = 0$. We argue that the graph output by Algorithm 1 satisfies $L_{H^\ell} \approx_\epsilon L_\ell$ below, together with the same argument for the inductive step.

Inductive step: $\ell - 1 \to \ell$. As per line 6 of Algorithm 1 we set $\widetilde{K} = \frac{1}{2(1+\epsilon)}$ SPARSIFY($\Pi^{\leq \ell-1} B, \ell - 1, \epsilon$), therefore the corresponding Laplacian for this call is $L_{\ell-1}$. By the inductive hypothesis SPARSIFY($\Pi^{\leq \ell-1} B, \ell - 1, \epsilon$) returns an ϵ-sparsifier of $L_{\ell-1}$, so we have

$$(1-\epsilon) \cdot L_{\ell-1} \preceq_r 2(1+\epsilon)\widetilde{K} \preceq_r (1+\epsilon) \cdot L_{\ell-1}. \tag{1.16}$$

Moreover, by Lemma A.0.1, we have

$$\frac{1}{2} \cdot L_\ell \preceq \frac{1}{2} \cdot L_{\ell-1} \preceq L_\ell. \tag{1.17}$$

Putting (1.16) and (1.17) together we get

$$\frac{1-\epsilon}{2(1+\epsilon)} \cdot L_\ell \preceq_r \widetilde{K} \preceq_r L_\ell, \tag{1.18}$$

which implies for $\epsilon \leq 1/5$ that

$$\frac{1}{3} \cdot L_\ell \preceq_r \widetilde{K} \preceq_r L_\ell. \tag{1.19}$$

We thus have that for all values of ℓ the matrix $\widetilde{K}$ defined at the beginning of Algorithm 1 is a 3-spectral sparsifier of $G^{\gamma(\ell)}$, assuming the inductive hypothesis for $\ell - 1$ (except for the base case case, where no

inductive hypothesis is needed). Consequently, for any pair of vertices (u,v) in the same connected component in L,

$$b_{uv}^\top L_\ell^+ b_{uv} \le b_{uv}^\top \widetilde{K}^+ b_{uv} \le 3 \cdot b_{uv}^\top L_\ell^+ b_{uv} \tag{1.20}$$

For the rest of the proof, we let $L := L_\ell$ for simplicity. We now show that the rest of the algorithm constructs an ϵ-sparsifier for L by sampling each edge e with some probability at least $\min\{1, R_{uv}^L \log(n)/\epsilon^2\}$ and giving it weight inverse proportional to the probability. This will indeed give us an ϵ-sparsifier due to Theorem A.0.1. In particular, this probability will be the following: For edges e in the appended complete graph γI the probability is 1. For an edge e in the original graph G we define the variable p'_e, as in line 18 of Algorithm 2, to be $\frac{5}{4} \cdot c_2 \cdot ||Mb_e||_2^2 \cdot \log(n)/\epsilon^2$, and we define p_e to be $\min\{1, p'_e\}$. Let s_e be the integer such that $p'_e \in (2^{-s_e-1}, 2^{-s_e}]$. Note that then $p_e \in (2^{-s_e^+-1}, 2^{-s_e^+}]$. Our probability for sampling an edge e of the original graph will be $2^{-s_e^+}$, which is less than $\min\{1, c_2 \cdot R_e^L \log(n)/\epsilon^2\}$, as required by Theorem A.0.1.

Consider the conditions of Lemma 1.3.2.

1. $\frac{1}{3} \cdot L_\ell \preceq_r \widetilde{K} \preceq_r L_\ell$ is satisfied as shown above.
2. Note that $R_{uv}^{\widetilde{K}} = ||\widetilde{W}^{1/2} \widetilde{B} \widetilde{K}^+ b_u - \widetilde{W}^{1/2} \widetilde{B} \widetilde{K}^+ b_v||_2^2$ so we can use the Johnson-Lindenstrauss lemma to approximate $R_{uv}^{\widetilde{K}}$ using a smaller matrix. In lines 10 and 9 we use the exact construction of Lemma A.0.3 with q being large enough for parameters $\epsilon = 1/5$ and $\beta = 6$. Therefore, $R_{uv}^{\widetilde{K}} \le \frac{5}{4}||M(\chi_u - \chi_v)||_2^2 \le \frac{3}{2} R_{uv}^{\widetilde{K}}$ is satisfied with high probability, by Lemma A.0.3.

Thus by Lemma 1.3.2 if edge e is sampled in $B_{s_e^+}^\ell$ then RECOVEREDGES$(\Pi_{s_e^+}^\ell B_{s_e^+}^\ell, M, \widetilde{K}^+, s_e, q, \epsilon)$ will recover e with high probability in line 14 of Algorithm 1. It will then be given the required weight $(2^{s_e^+})$. Note that e will not be recovered in any other call of RECOVEREDGES, that is when $s \neq s_e$. Note also, that $2^{-s_e^+}$ is indeed an upper bound on $\min\{1, c_2 \cdot R_e^L \cdot \log(n)/\epsilon^2\}$, and within constant factor of it. Therefore, by Theorem A.0.1, the resulting graph will be an ϵ-spectral sparsifier of G^γ, and it will be $O(n \cdot \text{polylog}(n)/\epsilon^2)$-sparse (disregarding the regularization). □

1.3.3 Maintenance of sketches

Note that Algorithm 2 takes sketch ΠB as input. More precisely, Π is a concatenation of HEAVYHITTER sketch matrices composed with sampling matrices, indexed by sampling rate s and regularization level ℓ. In particular, for all s and ℓ let B_s^ℓ be a row-sampled version of B at rate 2^{-s}. Then Π_s^ℓ is a HEAVYHITTER sketch drawn from the distribution from Lemma A.0.2 with parameter $\eta = \frac{1}{2} \cdot \frac{1}{C \cdot q^3} \cdot \sqrt{\frac{\epsilon^2}{\log n}}$. Note that the matrices $\left(\Pi_s^\ell\right)_{s,\ell}$ are independent and identically distributed. We then maintain $\Pi_s^\ell B_s^\ell$ for all s and ℓ. We define

$$\Pi^\ell B = \Pi_0^\ell B_0^\ell \oplus \ldots \oplus \Pi_{10\log n}^\ell B_{10\log n}^\ell,$$

where $\oplus$ denotes concatenation of rows. We let $\Pi^{\le \ell}$ denote $\Pi^0 \oplus \ldots \oplus \Pi^\ell$, and let Π denote $\Pi^{\le d+1}$ to simplify notation. Thus, the algorithm maintains ΠB throughout the stream. We maintain ΠB by maintaining each $\Pi_s^\ell B_s^\ell$ individually. To this end we have for each s and ℓ an independent hash function h_s^ℓ mapping $\binom{V}{2}$

to $\{0,1\}$ independently such that $\mathbb{P}(h_s^\ell(u,v)=1)=2^{-s}$. Then when an edge insertion or deletion, $\pm(u,v)$, arrives in the stream, we update $\Pi_s^\ell B_s^\ell$ by $\pm\Pi_s^\ell \cdot b_{uv} \cdot h_s^\ell(u,v)$.

Overall, the number of random bits needed for all the matrices in an invocation of Algorithm 2 is at most $R=\widetilde{O}(n^2)$, in addition to the random bits needed for the recursive calls. To generate matrix Π we use the fast pseudo random numbers generator from Theorem 1.3.5 below:

Theorem 1.3.5. *[28] For any constants $q,c>0$, there is an explicit pseudo-random generator (PRG) that draws on a seed of $O(S\,\mathrm{polylog}(S))$ random bits and can simulate any randomized algorithm running in space S and using $R=O(S^q)$ random bits. This PRG can output any pseudorandom bit in $O(\log^{O(q)} S)$ time and the simulated algorithm fails with probability at most S^{-c} higher than the original.*

Observe that the space used by Algorithm 2 is $s=\widetilde{O}(n)$ in addition to the space used by the recursive calls. Since $R=O(n^2)$, we have $R=O(s^2)$. Therefore, by Theorem 1.3.5 we can generate seed of $O(s\cdot\mathrm{poly}(\log s))$ random bits in $O(s\cdot\mathrm{poly}(\log s))$ time that can simulate our randomized algorithm.

Also, note that the random matrix $Q\in\mathbb{R}^{\Theta(\log n)\times\binom{n}{2}}$ for JL (line 9 of Algorithm 2) can be generated using $O(\log n)$-wise independent hash functions.

1.3.4 Proof of Theorem 1.1.1

Proof of Theorem 1.1.1:

Correctness of Algorithm 2 is proved in Theorem 1.3.4. It remains to prove space and runtime bounds.

Run-time and space analysis. We will prove that one call of SPARSIFY in Algorithm 1 requires $\widetilde{O}(n)$ time and space, discounting the recursive call, where n is the size of the vertex set of the input graph. Consider first lines 9 and 11, and note that the random matrix $Q\in\mathbb{R}^{\Theta(\log n)\times\binom{n}{2}}$ for JL (line 9 of Algorithm 2) can be generated using $O(\log n)$-wise independent hash functions, resulting in $\mathrm{poly}(\log n)$ time to generate an entry of Q and $O(\log n)$ space. We then multiply $Q\widetilde{W}^{1/2}\widetilde{B}$ by $\widetilde{K}^+$ which amounts to solving $\Theta(\log n)$ Laplacian systems and can be done in $O(n\,\mathrm{polylog}\,n\cdot\epsilon^{-2})$ time, since $\widetilde{K}$ is $O(n\,\mathrm{polylog}(n)\cdot\epsilon^{-2})$ sparse, using any of a variety of algorithms in the long line of improvements in solving Laplacian systems [4, 36, 37, 64, 65, 38, 39, 31, 32, 40]. The resulting matrix, M, is again $\Theta(\log n\times n)$ and can be stored in $n\,\mathrm{polylog}\,n$ space. We note that the aforementioned Laplacian solvers provide approximate solutions with inverse polynomial precision, which is sufficient for application of the HEAVYHITTER sketch.

The for loops in both line 13 and line 6 iterate over only $\Theta(\log n)$ values. For all non-empty cells we iterate over all vertices in that cell, so overall, we iterate n times. The HEAVYHITTERS subroutine called with parameter $\eta=\epsilon/\,\mathrm{polylog}\,n$ returns by definition at most $\mathrm{polylog}\,n/\epsilon^2$ elements, so the for loop in line 17 is over $\mathrm{polylog}\,n/\epsilon^2$ iterations. In total this is $O(n\,\mathrm{polylog}\,n\cdot\epsilon^{-2})$ time and space as claimed.

To get an ϵ-sparsifier of the input graph G, we need only to run SPARSIFY$(\Pi^{\le d+1}B,d+1,\epsilon)$. Therefore chain of recursive calls will be $\Theta(\log(n))$ long, and the total run time will still be $\widetilde{O}(n\epsilon^{-2})$. □

2 Graph Spanners by Sketching in Dynamic Streams and the Simultaneous Communication Model

This chapter is based on a joint work with Arnold Filtser and Michael Kapralov. It has been accepted to the ACM-SIAM Symposium on Discrete Algorithms [66, SODA].

2.1 Introduction

Graph sketching, introduced by [67] in an influential work on graph connectivity in dynamic streams has been a de facto standard approach to constructing algorithms for dynamic streams, where the algorithm must use a small amount of space to process a stream that contains both edge insertions and deletions. The main idea of [67] is to represent the input graph by its edge incident matrix, and applying classical linear sketching primitives to the columns of this matrix. This approach seamlessly extends to dynamic streams, as by linearity of the sketch one can simply subtract the updates for deleted edges from the summary being maintained: a surprising additional benefit is the fact that such a sketching solution is trivially parallelizable: since the sketch acts on the columns of the edge incidence matrix, the neighborhood of every vertex in the input graph is compressed independently. In particular, this yields efficient protocols in the *simultaneous communication* model, where every vertex knows its list of neighbors, and must communicate a small number of bits about this neighborhood to a coordinator, who then announces the answer. Surprisingly, several fundamental problems such as connectivity [67], cut [68] and spectral sparsification [69, 7, 14] admit sketch based simultaneous communication protocols with only polylogarithmic communication overhead per vertex, which essentially matches existentially optimal bounds.[1] The situation is entirely different for the problem of approximating the shortest path metric of the input graph: it is not known whether existentially best possible space vs approximation quality tradeoffs can be achieved using a linear sketch. This motivates the main question that we study:

[1] There is some overhead to using linear sketches, but it is only polylogarithmic in the number of vertices in the graph – see [15].

What are the optimal space/stretch/pass tradeoffs for approximating the shortest path metric using a linear sketch?

Sketching and dynamic streams. Sketching is the most popular tool for designing algorithms for the dynamic streaming model. Sketching solutions have been recently constructed for many graph problems, including spanning forest computation [1], cut and spectral sparsifiers [69, 7, 14], spanner construction [68, 19], matching and matching size approximation [20, 21], sketching the Laplacian [70, 23] and many other problems. Also, results showing universality of sketching for this application are known, at least under some restrictions on the stream. The result of [71] shows such an equivalence under the assumption that the stream length is at least doubly exponential in the size of the graph. The assumption on the stream length was significantly relaxed for *binary* sketches, i.e., sketches over $\mathbb{GF}_2$, by [72, 73]. Very recently, it has been shown [74] that lower bounds on stream length are crucial for such universality results: the authors of [74] exhibit a problem with a sketching complexity, which is polynomial in the input size, that can be solved in polylogarithmic space on a short dynamic stream.

Spanners in the sketching model. A subgraph $H = (V, E)$ of a graph $G = (V, E)$ is a t-spanner of G if for every pair $u, v \in V$ one has

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

where d_G stands for the shortest path metric of G and d_H for the shortest path metric of H. We assume in this paper that the input graph is unweighted, as one can reduce to this case using standard techniques at the expense of a small loss in space complexity.[2] For every integer $k \geq 1$, every graph $G = (V, E)$ with n vertices admits a $(2k-1)$-spanner with $O(n^{1+1/k})$ edges, which is optimal assuming the Erdős girth conjecture. The greedy spanner [77, 78], which is sequential by nature, obtain the optimal number of edges. The celebrated algorithm of Baswana and Sen [79] obtains $(2k-1)$-spanner with $\widetilde{O}(n^{1+1/k})$ edges. This algorithm consists of a sequence of k clustering steps, and as observed by [68], can be implemented in k passes over the stream using the existentially optimal $\widetilde{O}(n^{1+1/k})$ space. A central question is therefore whether it is possible to achieve the existentially optimal tradeoff using fewer rounds of communication, and if not, what the optimal space vs stretch tradeoff is for a given number of round of communication. Prior to our work this problem was studied in [68] and [19]. The former showed how to construct a $(k^{\log_2 5} - 1)$-spanner in $\log_2 k$ passes using space $\widetilde{O}(n^{1+1/k})$, and the latter showed how to construct a $(2^k - 1)$-spanner in two passes and $\widetilde{O}(n^{1+1/k})$ space. In a single pass, the previously best known algorithm which uses $n^{1+o(1)}$ space is simply to construct a spanning tree, guaranteeing distortion $n-1$. Thus, our first question is:

[2] Specifically, one can partition the input edges into geometric weight classes and run our sketch based algorithm on every class, paying a multiplicative loss in space bounded by the log of the ratio of the largest weight to the smallest weight. See also [75, 76].

In a single pass in the dynamic semi streaming model using $\widetilde{O}(n)$ space, is it possible to construct an $o(n)$ spanner?

We prove the following theorem in Section 2.4, as a corollary we obtain a positive answer to the question above (as spectral sparsifier can be computed in a single dynamic stream pass [7]).

Theorem 2.1.1. *Let $G = (V, E)$ be an undirected, unweighted graph. For a parameter $\varepsilon \in (0, \frac{1}{18}]$, suppose that H is a $(1 \pm \varepsilon)$-spectral sparsifier of G. Then $\widehat{H}$ is an $\tilde{O}(n^{\frac{2}{3}})$-spanner of G, where $\widehat{H}$ is unweighted version of H.*

Corollary 2.1.1. *There exists an algorithm that for any n-vertex unweighted graph G, the edges of which arrive in a dynamic stream, using $\tilde{O}(n)$ space, constructs a spanner with $O(n)$ edges and stretch $\widetilde{O}(n^{\frac{2}{3}})$ with high probability.*

Additionally, for the same setting, using similar techniques, we prove stretch $\widetilde{O}(\sqrt{m})$ (see Theorem 2.4.1).

One might think that the polynomial stretch is suboptimal, but we conjecture that this is close to best possible, and provide a candidate hard instance for a lower bound in Section B.1. Specifically,

Conjecture 2.1.1. *Any linear sketch from which one can recover an $n^{2/3-\Omega(1)}$-spanner with probability at least 0.9 requires $n^{1+\Omega(1)}$ space.*

More generally, we give the following trade off between stretch and space in a single pass:

Corollary 2.1.2. *Consider an n-vertex unweighted graph G, the edges of which arrive in a dynamic stream. For every parameter $\alpha \in (0, 1)$, there is an algorithm using $\widetilde{O}(n^{1+\alpha})$ space, constructs a spanner with stretch $\widetilde{O}(n^{\frac{2}{3}(1-\alpha)})$ with high probability.*

Similarly, for the same setting, we prove stretch $\widetilde{O}(\sqrt{m} \cdot n^{-\alpha})$ (see Theorem 2.4.3).

Simultaneous communication model. We also consider the related simultaneous communication model[3], which we now define. In the *simultaneous communication model* every vertex of the input graph $G = (V, E), |V| = n$, knows its list of neighbors (so that every $e = (u, v) \in E$ is known to both u and v), and all vertices have a source of shared randomness. Communication proceeds in rounds, where in every round the players simultaneously post short messages on a common board for everyone to see (note that equivalently, one could consider a coordinator who receives all the messages in a given round, and then posts a message of unbounded length on the board). Note that a given player's message in any given round may only depend on their input and other players' messages in previous rounds. The content of the board at the end of the communication protocol must reveal the answer with high constant probability. The *cost* of a protocol in the simultaneous communication model is the length of the longest message communicated by any player.

[3]This model has also been referred to as **distributed sketching** in the literature (see e.g., [15]).

Sketching algorithms for dynamic connectivity and cut/spectral approximations based on the idea of applying a sketch to the edge incidence matrix of the input graph [67, 7, 14] immediately yield efficient single pass simultaneous communication protocols with only polylogarithmic message length. We note, however, that existing sketch based algorithms for spanner construction (except for our result in Corollary 2.1.1 and the trivial k-pass implementation of the algorithm of Baswana and Sen [79]) **do not** yield low communication protocols. This is because they achieve reductions in the number of rounds by performing some form of leader election and amortizing communication over all vertices. To illustrate the difference between dynamic streaming and simultaneous communication model, consider the following artificial problem. Suppose that we are given a graph with all but $\sqrt{n}$ isolated vertices, and the task is to recover the induced subgraph. Then using sparse recovery we can recover all the edges between the vertices using a single pass over a dynamic stream of updates. However, it is clear from information theoretic considerations that a typical vertex will need to communicate $\Omega(\sqrt{n})$ bits of information to solve this problem.

The main result of Section 2.5.1 is the following theorem.

Theorem 2.1.2. *For any integer $g \geq 1$, there is an algorithm (see Algorithm 3) that in g rounds of communication outputs a spanner with stretch* $\min\left\{\tilde{O}(n^{\frac{g+1}{2g+1}}),\ (12+o(1))\cdot n^{2/g}\cdot\log n\right\}$.

Note that when $g = 1$, the above theorem gives a $\tilde{O}(n^{2/3})$ approximation using polylogarithmic communication per vertex. We think that the $n^{2/3}$ approximation is likely best possible in polylogarithmic communication per vertex, and the same candidate hard instance from Section B.1 that we propose for Conjecture 2.1.1 can probably be used to obtain a matching lower bound. Analyzing the instance appears challenging due to the fact that every edge is shared by the two players – exactly the feature of our model that underlies our algorithmic results (this sharing is crucial for both connectivity and spectral approximation via sketches). This model bears some resemblance to the number-on-the-forehead (NOF) model in communication complexity (see, for example, [80], where a connection of this form was made formal, resulting in conditional hardness results for subgraph counting in data streams).

The proof of this theorem is presented in Section 2.5.1.

Additionally, in Subsection 2.5.2 we first provide a trade off between size of the communication per player and stretch in one round of communication.

Theorem 2.1.3. *There is an algorithm that in* 1 *round of communication, where each player communicates $\tilde{O}(n^{\alpha})$ bits, outputs a spanner with stretch*

$$\min\left\{\widetilde{O}(n^{(1-\alpha)\frac{2}{3}}),\ \widetilde{O}\left(\sqrt{m}\cdot n^{-\alpha}\right)\right\}.$$

Then, we also prove a similar trade off when more than one round of communication is allowed.

Theorem 2.1.4. *For any integer $g \geq 1$, there is an algorithm that in g rounds of communication, where each player communicates $\tilde{O}(n^{\alpha})$ bits, outputs a spanner with stretch*

$$\min\left\{(12+o(1))\cdot n^{(1-\alpha)\cdot\frac{2}{g}}\cdot\log n\,,\ \widetilde{O}\left(n^{\frac{(g+1)(1-\alpha)}{2g+1}}\right)\right\}.$$

Related work. Streaming algorithms are well-studied with too many results to list and we refer the reader to [47, 16] for a survey of streaming algorithms. The idea of linear graph sketching was introduced in a seminal paper of Ahn, Guha, and McGregror [1]. An extension of the sketching approach to hypergraphs were presented in [48]. The simultaneous communication model has also been used for lower bounding the performance of sketching algorithms – see, e.g. [20, 81].

Spanners are a fundamental combinatorial object. They have been extensively studied and have found numerous algorithmic applications. We refer to the survey [82] for an overview. The most relevant related work is on insertion only streams [83, 84] where the focus is on minimizing the processing time of the stream, and dynamic algorithms, where the goal is to efficiently maintain a spanner while edges are continuously inserted and deleted [83, 85, 86].

2.2 Preliminaries

All the logarithms in the paper are in base 2. We use $\tilde{O}$ notation to suppress constants and poly-logarithmic factors in n, that is $\widetilde{O}(f) = f \cdot \text{polylog}(n)$.

We consider undirected, graphs $G = (V, E)$, with a weight function $w : E \to \mathbb{R}_{\geq 0}$. If we say that a graph is unweighted, we mean that all the edges have unit weight. $\widehat{G} = (V, E, \mathbb{1}_E)$ denotes the unweighted version of G, i.e. the graph G where all edge weights are changed to 1. Sometimes we abuse notation and write G instead of E. Given two subsets $X, Y \subseteq V$, $E_G(X, Y)$ is the set of edges from X to Y, $w_G(X, Y)$ denotes the total weight of edges in $E_G(X, Y)$ (number if G is unweighted). We sometimes abuse notation and write instead $E_G(X \times Y)$ and $w_G(X \times Y)$ (respectively). For a subset of vertices $A \subseteq V$, let $G[A]$ denote the induced graph on A.

Let d_G denote the shortest path metric in G. A subgraph H of G is a *t-spanner* of G if for every $u, v \in V$, $d_H(u, v) \leq t \cdot d_G(u, v)$ (note that as H is a subgraph of G, necessarily $d_G(u, v) \leq d_H(u, v)$). Following the triangle inequality, in order to prove that H is a t-spanner of G it is enough to show that for every edge $(u, v) \in E$, $d_H(u, v) \leq t \cdot d_G(u, v)$.

For an unweighted graph $G = (V, E)$, such that $|V| = n$ and $|E| = m$, let $B_G \in \mathbb{R}^{m \times n}$ denote the vertex edge incidence matrix. The Laplacian matrix of G is defined as $L_G := B_G^\top B_G$. Similarly, for a weighted graph $H = (V, E, w)$, we let $W \in \mathbb{R}^{m \times m}$ be the diagonal matrix of the edge weights. The Laplacian of the graph H is defined as $L_H := B_H^\top W B_H$. $H \preceq G$ denotes that for every $\vec{x} \in \mathbb{R}^n$, $\vec{x}^t L_H \vec{x} \leq \vec{x}^t L_G \vec{x}$. We say that a graph H is $(1 \pm \epsilon)$-*spectral sparsifier* of a graph G, if

$$(1 - \epsilon) H \preceq G \preceq (1 + \epsilon) H .$$

Fact 2.2.1. *Suppose that a graph H is a $(1 \pm \epsilon)$-spectral sparsifier of a graph G, then H is a $(1 \pm \epsilon)$-cut sparsifier of G, i.e., for every set of vertices $S \subset V$, we have*

$$(1 - \epsilon) \cdot w_H(S, V \setminus S) \;\leq\; w_G(S, V \setminus S) \;\leq\; (1 + \epsilon) \cdot w_G(S, V \setminus S) .$$

For any Laplacian matrix L_G, we denote its Moore-Penrose pseudoinverse by L_G^+. For any pair of vertices $u, v \in V$, we denote their indicator vector by $b_{uv} = \chi_u - \chi_v$, where $\chi_u \in \mathbb{R}^n$ is the indicator vector of u, i.e., the entry corresponding to u is $+1$ and all other entries are zero. Also, for any edge $e = (u, v)$, we define its indicator vector as $b_e := b_{uv}$. We also define effective resistance of a pair of vertices $u, v \in V$ as

$$R_{uv}^G := b_{uv}^\top L_G^+ b_{uv}.$$

Fact 2.2.2. *Given a $(1 \pm \epsilon)$-spectral sparsifier H of a G, for every $u, v \in V$ it holds that*

$$(1-\epsilon) R_{uv}^G \;\le\; R_{uv}^H \;\le\; (1+\epsilon) R_{uv}^G.$$

The following fact is a standard fact about effective resistances (see e.g., [30])

Fact 2.2.3. *In every n vertex graph $G = (V, E, w)$ it holds that $\sum_{e \in E} w_e R_e^G \;\le\; n - 1$.*[4]

Dynamic streams. In dynamic streams, there is a fixed set V of n vertices, unweighted edges arrive in a streaming fashion, where they are both inserted and deleted.

ℓ_0-samplers.: Given integer vector in $\mathbb{R}^n$ in a dynamic stream, using $s \cdot \text{polylog}(n)$ space, we can sample s different non-zero entries. In particular if the vector is s-sparse, we can reconstruct it. Furthermore, given a stream of edges in an n-vertex graph G, using $s \cdot \text{polylog}(n)$ samplers per vertex, we can create a subgraph $\tilde{G}$ of H where each vertex has either at least s edges, or has all its incident edges from G. This samplers are linear, therefore if we sum up the samplers of S vertices, we can sample an outgoing edge.

Consider a vector $\vec{v} \in \mathbb{R}^n$, given a subset $A \subseteq [n]$ of coordinates, we denote by $\vec{v}[A]$ the restriction of $\vec{v}$ to A.

Lemma 2.2.1. *Consider a vector $\vec{v} \in \mathbb{R}^n$ that arrives in a dynamic stream via coordinate updates. The coordinates $[n]$ are partitioned into subsets $A_1, A_2, \ldots, A_r$ (the space required to represent this partition is negligible). Let $\mathscr{I} = \left\{ i \mid \vec{v}[A_i] \neq \vec{0} \right\}$ be the indices of the coordinate sets on which $\vec{v}$ is not zero. Given $A_1, A_2, \ldots, A_r$ and a parameter $s > 0$, and a guarantee that $|\mathscr{I}| \le s$, using $s \cdot \text{polylog}(n)$ space, one can design a sketching algorithm recovering a set $S \subseteq [n]$ such that*

- *For every $j \in S$, $\vec{v}_j \neq 0$.*
- *For every $i \in \mathscr{I}$, $A_i \cap S \neq \emptyset$.*

The proof uses a technique commonly used in sketching literature, and is given in Section B.2.1 for completeness.

Lemma 2.2.2. *[Edge recovery] Consider an unweighted, undirected graph $G = (V, E)$ that is received in a dynamic stream. Given $A, B \subseteq V$ such that $A \cap B = \emptyset$, one can design a sketching algorithm that using $\text{polylog}(n)$ space in a single pass over the stream, with probability $1/\text{poly}(n)$, can either recover an edge*

[4] If graph G is connected, then the inequality is satisfied by equality.

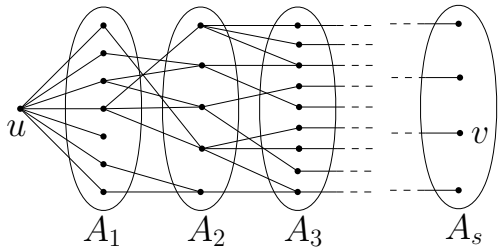


Figure 2.1: An illustration of the graph discussed in Section 2.3.

between A to B, or declare that there is no such edge.
Further, provided that there are at most m edges in $A \times B$, using $m \cdot \text{polylog}(n)$ space, with probability $1/poly(n)$ we can recover them all.

The proof is using the same techniques as in the proof of Lemma 2.2.1 and is deferred to Section B.2.1.

2.3 Technical Overview

We consider an n vertex unweighted graph $G = (V, E)$.

Spectral sparsifiers are spanners (Section 2.4). The technical part of the paper begins by proving the following fact: consider a spectral sparsifier H of G. Consider an edge $(u, v) \in E$. Denote the distance between its endpoints in $\widehat{H}$ by $d_{\widehat{H}}(u, v) = s$. Divide the vertices V into the BFS layers w.r.t. u in $\widehat{H}$. That is, A_i is the set of all vertices at distance i from u in $\widehat{H}$. In particular $v \in A_s$. See illustration in Figure 2.1. Let $W_i^G = w_G(A_i \times A_{i+1})$ be the total weight of the edges in $E_G(A_i, A_{i+1})$. Similarly $W_i^H = w_H(A_i \times A_{i+1})$. Let H' be the graph created from H by contracting all the vertices in each set A_i into a single vertex. The rough intuition is the following:

$$1 \overset{(a)}{\geq} R^G_{u,v} \overset{(b)}{\gtrsim} R^H_{u,v} \overset{(c)}{\geq} R^{H'}_{u,v} \overset{(d)}{=} \sum_{i=0}^{s-1} \frac{1}{W_i^H} \overset{(*)}{\approx} \sum_{i=0}^{s-1} \frac{1}{W_i^G} \overset{(e)}{\geq} \sum_{i=0}^{s-1} \frac{1}{|A_i||A_{i+1}|} \overset{(f)}{\geq} \Omega\left(\frac{s^3}{n^2}\right). \tag{2.1}$$

Here (a) follows as the effective resistance between the endpoints of an edge is at most 1. (b) as H is a spectral sparsifier of G. (c) as the effective resistance can only reduce by contracting vertices. (d) as H' is a path graph. (e) as G is unweighted and thus W_i^G is bounded by the number of edges in $A_i \times A_{i+1}$. And (f) as $\sum_i |A_i| \leq n$ and the function $\sum_{i=0}^{s-1} \frac{1}{|A_i||A_{i+1}|}$ is minimized when $|A_i| = \Omega(\frac{n}{s})$ for all i. The tricky part is the rough equality (*). Note that if (2.1) holds, it will follow that $s = O(n^{\frac{2}{3}})$, implying the desired stretch.

While H is a spectral sparsifier of G, W_i^G does not represent the size of a cut in G. This is as there might be edges in G crossing from A_i to $\cup_{j>i+1} A_j$, or from A_{i+1} to $\cup_{j<i} A_j$. Thus a priori there is no reason to expect that W_i^H will approximate W_i^G. Interestingly, we were able to show that $W_i^G = W_i^H \pm \epsilon \cdot (W_{i-1}^H + W_i^H + W_{i+1}^H)$. That is, while we are not able to bound $|W_i^G - W_i^H|$ using the standard factor $\epsilon \cdot W_i^H$, we can bound this error once we take into account also the former and later cuts in the BFS order! We use this fact to show that for most of the indices i, $W_i^H \leq |A_i||A_{i+1}|$. The desired bound follows. See proof of Theorem 2.1.1 for more details.

Next, using similar analysis we show that in case where the graph G has m edges, the stretch of $\widehat{H}$ is bounded by $O(\sqrt{m})$ (see Theorem 2.4.1). Suppose that $d_{\widehat{H}}(u,v) = s$. Intuitively, following (2.1), as $\sum_i W_i^G \leq m$, it follows that $1 \geq R_{u,v}^G \gtrsim \sum_{i=0}^{s-1} \frac{1}{W_i^G} = \Omega\left(\frac{s^2}{m}\right)$ (as $\sum_{i=0}^{s-1} \frac{1}{W_i^G}$ us minimized when all W_i^G's are equal), implying $s = O(\sqrt{m})$. Both bounds ($O(n^{\frac{2}{3}})$ and $O(\sqrt{m})$) are tight. Essentially, we construct the exact instance tightening all the inequalities in (2.1). That is a graph with $\widetilde{\Theta}(n^{\frac{2}{3}})$ layers, each one containing $\widetilde{\Theta}(n^{\frac{1}{3}})$ vertices, and all possible edges between layers (see Subsection 2.4.2).

In Section 2.4.3, we show that using $\widetilde{O}(n^{1+\alpha})$ space (instead of $\widetilde{O}(n)$), the stretch can be reduced to

$$\min\{\widetilde{O}(n^{\frac{2}{3}(1-\alpha)}), \widetilde{O}(\sqrt{m} \cdot n^{-\alpha})\}.$$

The idea is the following: randomly partition the graph G into $\widetilde{O}(n^{2\alpha})$ induced subgraphs $G_1, G_2, \ldots$, such that each G_i contains $O(n^{1-\alpha})$ vertices, and every pair of vertices u, v belong to some G_i. Furthermore, the (expected) number of edges in each G_i is $m \cdot n^{-2\alpha}$. Next, we construct a spectral sparsifier for each graph G_i and take their union as our spanner. The stretch gurantee follows (see Theorem 2.4.2, Theorem 2.1.2 and Theorem 2.4.3).

Simultaneous communication model (Section 2.5). In a single pass, one can construct a spectral sparsifier and therefore obtain the exact same results as in the streaming model. However, as opposed to streaming, no known approach can reduce the stretch in less than logarithmic number of rounds. We propose a natural peeling algorithm (see Algorithm 3). Denote $G_1 = G$. Given a desired stretch parameter t, the algorithm computes a spectral sparsifier H_1, and removes all the satisfied edges $(u,v) \in E$ where $d_{\widehat{H}_1}(u,v) \leq t$, to obtain a graph G_2. Generally, in the i'th round the algorithm computes a spectral sparsifier H_i for the graph G_i, and removes all the satisfied edges to obtain G_{i+1}. This procedure continues until all the edges are satisfied (that is $G_{i+1} = \emptyset$). The resulting spanner is $\widehat{H} = \cup_i \widehat{H}_i$ the union of (the unweighted version of) all the constructed sparsifiers. Notably, for every parameter $t \geq 1$ the algorithm will eventually halt, and return a t-spanner. The arising question is, how many rounds are required to satisfy a specific parameter t?

We show that this procedure will halt after g steps for

$$t \geq \min\{\tilde{O}(n^{\frac{g+1}{2g+1}}),\ (12 + o(1)) \cdot n^{2/g} \cdot \log n\}$$

(see Theorem 2.1.2). Interestingly, in $g = \log n$ rounds we can obtain stretch $O(\log n)$, which is asymptotically optimal. That is, we present a completely new construction for a $O(\log n)$-spanner with $\tilde{O}(n)$ edges. Interestingly, there are constructions of spectral sparsifiers which are based on taking a union of poly-logarithmically many $O(\log n)$-stretch spanners (see [87, 88]). In a sense, here we obtain the opposite direction. That is, by taking a union of $\log n$ sparsifiers, one can construct an $O(\log n)$ stretch spanner. That is, sparsifiers and spanners are much more related from what one may initially expect.

To show that the algorithm halts in g round for a specific t, we bound the number of edges in G_i, which eventually will lead us to conclusion that $G_{g+1} = \emptyset$:

- Set $t = \tilde{O}(n^{\frac{g+1}{2g+1}})$. Here the analysis is based on the effective resistance. Using (2.1), one can see that after the first round, G_2 will contain only edges with effective resistance at least $\Omega(\frac{t^3}{n^2})$ (in G). As the sum of all effective resistances is bounded by $n-1$, we conclude $|G_2| \leq \Omega(\frac{n^3}{t^3})$. In general, following the $O(\sqrt{m})$ upper bound on stretch, one can show that G_{i+1} contain only edges with effective resistance $\Omega(\frac{t^2}{|G_i|})$, implying $|G_{i+1}| \leq \frac{n}{t^2}|G_i|$. t is chosen so that $|G_g| \leq t^2$, hence a spectral sparsifier will have stretch at most $\sqrt{|G_g|} = t$ for all the edge, implying $G_{g+1} = \emptyset$.
- Set $t = O(n^{2/g} \cdot \log n)$. Here the analysis is based on low diameter decomposition. In general, for a weighted graph H and parameter $\phi = n^{-2/g}$, we construct a partition $\mathscr{C}$ of the vertices, such that each cluster $C \in \mathscr{C}$ has hop-diameter $O(\frac{\log n}{\phi}) = t$ (i.e. w.r.t. $\widehat{H}$), and the overall fraction of the weight of inter-cluster edges is bounded by ϕ. Following our peeling algorithm, when this clustering is preformed w.r.t. H_i, G_{i+1} will contain only inter-cluster edges from G_i. As H_i is a spectral sparsifier of G_i, the size of all cuts are preserved. It follows that $|G_{i+1}| \lesssim \phi \cdot |G_i|$. In particular, in $\log_{\frac{1}{\phi}} |G| \leq g$ rounds, no edges will remain.
  Interestingly, for this analysis to go through it is enough that each H_i will be a cut sparsifier of G_i, rather than a spectral sparsifier. Oppositely, a single cut sparsifier H of G can have stretch $\tilde{\Omega}(n)$ (see Remark 2.4.1).

Next, similarly to the streaming case, we show that if each player can communicate a message of size $\widetilde{O}(n^{\alpha})$ in each round, then we can construct a spanner with stretch $\min\{\widetilde{O}(n^{\frac{2}{3}(1-\alpha)}), \widetilde{O}(\sqrt{m} \cdot n^{-\alpha})\}$ in a single round, or stretch $\min\left\{(12+o(1)) \cdot n^{(1-\alpha)\cdot\frac{2}{g}} \cdot \log n\,,\ \widetilde{O}\left(n^{\frac{(g+1)(1-\alpha)}{2g+1}}\right)\right\}$ in g rounds (see Theorem 2.1.3 and Theorem 2.1.4). The approach is the same as in the streaming case, and for the most part, the analysis follows the same lines. However, the single round $\widetilde{O}(\sqrt{m} \cdot n^{-\alpha})$ bound is somewhat more involved. Specifically, in the streaming version we've made the assumption that $m \geq n^{1+\alpha}$, as otherwise, using sparse recovery we can restore the entire graph. Unfortunately, sparse recovery is impossible here. Instead, we show that in a single communication round we can partition the vertex set V into V_1, V_2, such that all the incident edges of V_1 are restored, while the minimum degree in $G[V_2]$ is at least n^{α}. The rest of the analysis goes through.

2.4 Spectral Sparsifiers are Spanners

In this section, we show that spectral sparsifiers can be used to achieve low stretch spanners in one pass over the stream. Our algorithm works as follows: first, given a graph $G = (V, E)$, it generates a (possibly weighted) spectral sparsifier H of G, using the sketches which can be stored in $\tilde{O}(n)$ space [7, 89, 14]. Then, the weights of all edges are set to be equal to 1. We show that the resulting graph $\widehat{H}$ is a $\tilde{O}(n^{\frac{2}{3}})$-spanner of the original graph.

Theorem 2.1.1. *Let $G = (V, E)$ be an undirected, unweighted graph. For a parameter $\varepsilon \in (0, \frac{1}{18}]$, suppose that H is a $(1 \pm \varepsilon)$-spectral sparsifier of G. Then $\widehat{H}$ is an $\tilde{O}(n^{\frac{2}{3}})$-spanner of G, where $\widehat{H}$ is unweighted version of H.*

As [7] constructed $(1 \pm \varepsilon)$-spectral sparsifier with $O(\frac{n}{\varepsilon^2})$ edges in a dynamic stream, by fixing $\varepsilon = \frac{1}{18}$, we conclude:

Corollary 2.1.1. *There exists an algorithm that for any n-vertex unweighted graph G, the edges of which arrive in a dynamic stream, using $\tilde{O}(n)$ space, constructs a spanner with $O(n)$ edges and stretch $\widetilde{O}(n^{\frac{2}{3}})$ with high probability.*

Proof of Theorem 2.1.1. By triangle inequality, it is enough to prove that for every edges $(u,v) \in E$, it holds that $d_{\widehat{H}}(u,v) = \tilde{O}(n^{\frac{2}{3}})$. Our proof strategy is as follows: consider a pair of vertices $u, v \in V$ such that $d_{\widehat{H}}(u,v) = s$. We will prove that $R^G_{u,v} \geq \tilde{\Omega}(\frac{s^3}{n^2})$. As for every pair of neighboring vertices it holds that $R^G_{u,v} \leq 1$, the theorem will follow.

Consider a pair of vertices $v, u \in V$ such that $d_{\widehat{H}}(v,u) = s$. We partition V to sets $A_0, A_1, \ldots, A_s$ where for $i < s$, $A_i = \{z \in V \mid d_{\widehat{H}}(v,z) = i\}$ are all the vertices at distance i from v in $\widehat{H}$. $A_s = \{z \in V \mid d_{\widehat{H}}(v,z) \geq s\}$ are all the vertices at distance at least s. Let $W^H_i = w_H(A_i \times A_{i+1})$ be the total weight in H (the weighted sparsifier) of all the edges between A_i to A_{i+1}. Similarly, set $W^G_i = w_G(A_i \times A_{i+1})$. We somewhat abused notation here, we treat non-existing edges as having weight 0, while all the edges in the unweighted graph G have unit weight. For simplicity of notation set also $W^H_{-1} = W^G_{-1} = W^H_s = W^G_s = 0$. Note that while W^H_i denotes the size if a cut in H, it does not correspond to a cut in G (as e.g. there might be edges from A_i to A_{i+2}). Thus, a priori there should not be a resemblance between W^G_i to W^H_i. Nevertheless, we show that W^H_i approximates W^G_i. However, the approximation will depend also on W^H_{i-1}, W^H_{i+1} rather than only on W^H_i.

Claim 2.4.1. *For every i, $W^H_i - \epsilon \cdot (W^H_{i-1} + W^H_i + W^H_{i+1}) \leq W^G_i \leq W^H_i + \epsilon \cdot (W^H_{i-1} + W^H_i + W^H_{i+1})$.*

Proof of Claim 2.4.1. For a fixed i, set

$$A_{<i} = A_0 \cup \cdots \cup A_{i-1} \qquad A_{>i+1} = A_{i+2} \cup \cdots \cup A_s$$

$$A_{\leq i} = A_0 \cup \cdots \cup A_i \qquad A_{\geq i+1} = A_{i+1} \cup \cdots \cup A_s$$

In addition we denote the weight of several edge sets as follows, (see Figure 2.2 for illustration)

$$\begin{array}{lll} a = w_G(A_i \times A_{i+1}) & b = w_G(A_i \times A_{>i+1}) & c = w_G(A_{<i} \times A_{i+1}) \\ d = w_G(A_{<i} \times A_{>i+1}) & e = w_G(A_{<i} \times A_i) & f = w_G(A_{i+1} \times A_{>i+1}) \end{array} \tag{2.2}$$

Similarly by replacing w_G with w_H in (2.2), we obtain the values a', b', c', d', e', f' (e.g. $a' = w_H(A_i \times A_{i+1})$). Note that by the definition of the sets $A_0, \ldots, A_s$, it holds that $b' = c' = d' = 0$. Using this notation, Claim 2.4.1 states that $a' - \epsilon \cdot (a' + e' + f') \leq a \leq a' + \epsilon \cdot (a' + e' + f')$.

Note that any $(1 \pm \varepsilon)$-spectral sparsifier is a $(1 \pm \varepsilon)$-cut sparsifier. Thus, as H is a $(1 \pm \varepsilon)$-spectral sparsifier of G, it preserves weights of all the cuts up to ϵ error factors. We derive the following inequalities:

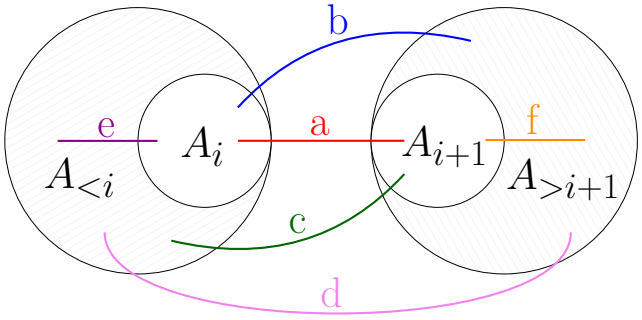


Figure 2.2: An illustration of the diffferent edges sets, the weight of which is denoted in (2.2). Note that $a = W_i^G$, $e = W_{i-1}^G$, and $f = W_{i+1}^G$.

$$\begin{array}{lcccll}
(1-\epsilon)a' & \leq & a+b+c+d & \leq & (1+\epsilon)a' & \text{By } (A_{\leq i}, A_{\geq i+1})\text{-cut} \\
(1-\epsilon)f' & \leq & b+d+f & \leq & (1+\epsilon)f' & \text{By } (A_{\leq i+1}, A_{>i+1})\text{-cut} \\
(1-\epsilon)e' & \leq & c+d+e & \leq & (1+\epsilon)e' & \text{By } (A_{<i}, A_{\geq i})\text{-cut} \\
(1-\epsilon)(a'+e'+f') & \leq & a+d+e+f & \leq & (1+\epsilon)(a'+e'+f') & \text{By } (A_{<i}\cup A_{i+1}, A_i \cup A_{>i+1})\text{-cut}
\end{array}$$

Or equivalently

$$\begin{array}{rcccc}
(1-\epsilon)a' & \leq & a+b+c+d & \leq & (1+\epsilon)a' \\
-(1+\epsilon)f' & \leq & -b-d-f & \leq & -(1-\epsilon)f' \\
-(1+\epsilon)e' & \leq & -c-d-e & \leq & -(1-\epsilon)e' \\
(1-\epsilon)(a'+e'+f') & \leq & a+d+e+f & \leq & (1+\epsilon)(a'+e'+f')
\end{array}$$

By summing up these 4 inequalities, and dividing by 2, we get

$$a' - \epsilon\cdot(a'+e'+f') \leq a \leq a' + \epsilon\cdot(a'+e'+f') .$$

The claim now follows. □

Our next goal is to bound $\sum_{i=0}^{s-1}\frac{1}{W_i^H}$, as this quantity lower-bounds the resistance between u and v in H. Since $\sum_{i=0}^{s}|A_i| = n$ and $W_i^G \leq |A_i|\cdot|A_{i+1}|$, one can bound $\sum_{i=0}^{s-1}\frac{1}{W_i^G}$ by $\Omega\left(\frac{s^3}{n^2}\right)$. However relating this quantity to the effective resistances in G is not as straightforward as one might expect.

Claim 2.4.2. $\sum_{i=0}^{s-1}\frac{1}{W_i^H} \geq \Omega\left(\frac{s^3}{n^2}\cdot\frac{\log^2\frac{1}{\epsilon}}{\log^2 n}\right)$.

Proof of Claim 2.4.2. For all $i \in [s]$, set $a_i = |A_i|$. Set

$$\alpha := 10\log_{\frac{1}{6\epsilon}} n^2 , \tag{2.3}$$

and

$$I := \left\{ i \in [s] \mid a_i \leq \frac{\alpha n}{s} \right\}.$$

It holds that $|I| \geq \left(1-\frac{1}{\alpha}\right)s+1$, as otherwise there are at least $\frac{s}{\alpha}$ indices i for which $a_i > \frac{\alpha n}{s}$, implying $\sum_i a_i > n$, a contradiction, since $A_0,\ldots,A_s$ forms a partition of V. Set

$$\widetilde{I} := \left\{ i \mid \text{such that } \forall j \text{ such that } |i-j| \leq \frac{\alpha}{10}, \text{ it holds that } j \in I \right\}.$$

Note that, since there are less than $\frac{s}{\alpha}$ indices i such that $i \notin I$, then there are less than $\frac{s}{\alpha} \cdot \frac{2\alpha}{10} \leq \frac{s}{5}$ indices out of $\widetilde{I}$, implying

$$\left|\widetilde{I}\right| \geq \frac{s}{2}\,. \tag{2.4}$$

Fix an index $i_0 \in \widetilde{I}$, we argue that $W^H_{i_0} \leq 2\left(\frac{\alpha n}{s}\right)^2$. For every index $j \in \left[i_0-\frac{\alpha}{10}, i_0+\frac{\alpha}{10}-1\right]$, it holds that $W^G_j = |E_G(A_j,A_{j+1})| \leq a_j \cdot a_{j+1} \leq \left(\frac{\alpha n}{s}\right)^2$. Assume for the sake of contradiction that $W^H_{i_0} > 2\left(\frac{\alpha n}{s}\right)^2$. We prove by induction that for $1 \leq j \leq \frac{\alpha}{10}$, there is an index i_j such that $|i_j - i_0| \leq j$ and $W^H_{i_j} > \frac{1}{(6\epsilon)^j}\left(\frac{\alpha n}{s}\right)^2$. For the base case, by Claim 2.4.1,

$$W^H_{i_0-1} + W^H_{i_0} + W^H_{i_0+1} \geq \frac{1}{\epsilon}\left(W^H_{i_0} - W^G_{i_0}\right) > \frac{1}{\epsilon}\left(2\left(\frac{\alpha n}{s}\right)^2 - \left(\frac{\alpha n}{s}\right)^2\right) = \frac{1}{\epsilon}\left(\frac{\alpha n}{s}\right)^2 .$$

The we can choose $i_1 \in \{i_0-1, i_0, i_0+1\}$ such that $W^H_{i_1} > \frac{1}{3\epsilon}\left(\frac{\alpha n}{s}\right)^2 > \frac{1}{6\epsilon}\left(\frac{\alpha n}{s}\right)^2$.
For the induction step, suppose that there is an index i_j such that $|i_j - i_0| \leq j < \frac{\alpha}{10}$ and $W^H_{i_j} > \frac{1}{(6\epsilon)^j}\left(\frac{\alpha n}{s}\right)^2$. As $|i_j - i_0| \leq \frac{\alpha}{10}-1$, it follows that $W^G_{i_j} \leq \left(\frac{\alpha n}{s}\right)^2$. Hence

$$W^H_{i_j-1} + W^H_{i_j} + W^H_{i_j+1} \geq \frac{1}{\epsilon}\left(W^H_{i_j} - W^G_{i_j}\right) \geq \frac{1}{\epsilon}\left(\frac{1}{(6\epsilon)^j}\left(\frac{\alpha n}{s}\right)^2 - \left(\frac{\alpha n}{s}\right)^2\right) > \frac{1}{2\epsilon}\cdot\frac{1}{(6\epsilon)^j}\left(\frac{\alpha n}{s}\right)^2 .$$

Thus there is an index $i_{j+1} \in \left\{i_j-1, i_j, i_j+1\right\}$ such that $W^H_{i_{j+1}} > \frac{1}{(6\epsilon)^{j+1}}\left(\frac{\alpha n}{s}\right)^2$, as required.

We conclude that,

$$W^H_{i_{\frac{\alpha}{10}}} > (6\epsilon)^{-\frac{\alpha}{10}}\left(\frac{\alpha n}{s}\right)^2 \overset{(2.3)}{\geq} n^2\left(\frac{\alpha n}{s}\right)^2 \geq n^2\,,$$

where the last inequality follows as $s \leq n$. This is a contradiction, as H is an $(1\pm\epsilon)$ spectral sparsifier of the unweighted graph G, where the maximal size of a cut is $\frac{n^2}{4}$. We conclude that for every $i \in \widetilde{I}$, it holds that $W^H_i \leq 2\left(\frac{\alpha n}{s}\right)^2$. The claim now follows as

$$\begin{aligned}
\sum_{i=0}^{s-1}\frac{1}{W^H_i} &\geq \left|\widetilde{I}\right| \cdot \frac{1}{2}\left(\frac{\alpha n}{s}\right)^{-2} \\
&\geq \frac{s^3}{4\alpha^2 n^2} && \text{By (2.4)} \\
&= \Omega\left(\frac{s^3}{n^2}\cdot\frac{\log^2\frac{1}{\epsilon}}{\log^2 n}\right) && \text{By (2.3)}
\end{aligned} \tag{2.5}$$

□

We are now ready to prove the theorem. Construct an auxiliary graph H' from H, by contracting all the vertices inside each set A_i, and keeping multiple edges. Note that by this operation, the effective resistance between u and v cannot increase. The graph H' is a path graph consisting of s vertices, where the conductance between the i'th vertex to the $i+1$'th is W_i^H. Using Claim 2.4.2, we conclude

$$\begin{aligned}
(1+\epsilon)R_{u,v}^G &\geq R_{u,v}^H && \text{By Fact 2.2.2} \\
&\geq R_{u,v}^{H'} && \text{As explained above} \\
&= \sum_{i=0}^{s-1} \frac{1}{W_i^H} && \text{Since } H' \text{ is a path graph} \\
&= \Omega\left(\frac{s^3}{n^2} \cdot \frac{\log^2 \frac{1}{\epsilon}}{\log^2 n}\right) && \text{By (2.5)} \qquad (2.6)
\end{aligned}$$

As u, v are neighbors in the unweighted graph G, it necessarily holds that $R_{u,v}^G \leq 1$, implying that $s = O\left(\left(n^2 \cdot \frac{\log^2 n}{\log^2 \frac{1}{\epsilon}}\right)^{\frac{1}{3}}\right) = \widetilde{O}\left(n^{\frac{2}{3}}\right)$. □

We state the following corollary, based on the last part of the proof of Theorem 2.1.1.

Corollary 2.4.1. *Let $G = (V, E)$ be an unweighted undirected graph, and let H be a $(1 \pm \epsilon)$-spectral sparsifier of G for some small enough constant ϵ. Also, let $\widehat{H}$ denote the unweighted H. If for a pair of vertices $u, v \in V$ we have $s := d_{\widehat{H}}(u, v)$, then*

$$R_{u,v}^G = \widetilde{\Omega}\left(\frac{s^3}{n^2}\right),$$

and

$$R_{u,v}^H = \widetilde{\Omega}\left(\frac{s^3}{n^2}\right).$$

2.4.1 Sparse graphs

Suppose we are guaranteed that the graph G we receive in the dynamic stream has eventually at most m edges. In Theorem 2.4.1 we show that the distortion guarantee of a sparsifier is at most $\widetilde{O}(\sqrt{m})$, and thus together with Theorem 2.1.1 it is $\widetilde{O}(\min\{\sqrt{m}, n^{\frac{2}{3}}\})$. Later, in Section2.5 we will use this to obtain a two pass algorithm in the simultaneous communication model with distortion $\widetilde{O}(n^{\frac{3}{5}})$.

Theorem 2.4.1. *Let $G = (V, E)$ be an undirected, unweighted such that $|V| = n$ and $|E| = m$. For a parameter $\varepsilon \in (0, \frac{1}{18}]$, suppose that H is a $(1 \pm \varepsilon)$-spectral sparsifier of G. Then $\widehat{H}$ is an $\widetilde{O}(\sqrt{m})$-spanner of G, where $\widehat{H}$ is the unweighted version of H.*

The proof follows similar lines to the proof of Theorem 2.1.1 and is deferred to Section B.2.2. Theorem 2.4.1 implies a streaming algorithm using space $\tilde{O}(n)$ that constructs a spanner with stretch $\tilde{O}(\sqrt{m})$. Notice that the number of edges m, does not need to be known in advance.

Similar to Corollary 2.4.1, using the last part of the proof of Theorem 2.4.1, we conclude the following:

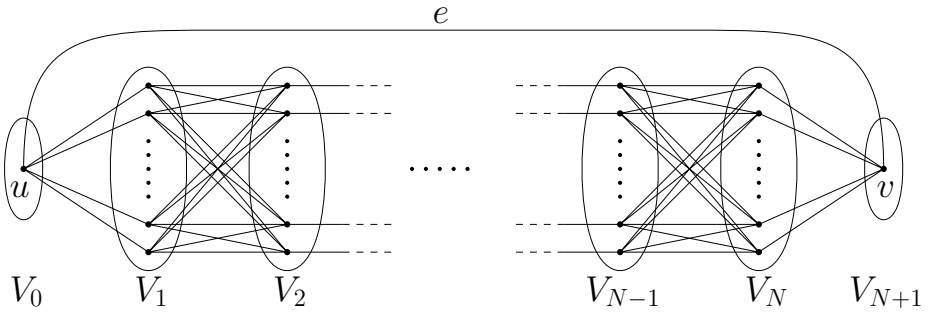


Figure 2.3: An illustration of the graph G constructed during the proof of Lemma 2.4.1.

Corollary 2.4.2. *Let $G = (V, E)$ be an unweighted undirected graph with $m = |E|$, and let H be a $(1 \pm \epsilon)$-spectral sparsifier of G for some small enough constant ϵ. Also, let $\widehat{H}$ denote the unweighted H. If for a pair of vertices $u, v \in V$ we have $s := d_{\widehat{H}}(u, v)$, then*

$$R^G_{u,v} = \widetilde{\Omega}\left(\frac{s^2}{m}\right),$$

and

$$R^H_{u,v} = \widetilde{\Omega}\left(\frac{s^2}{m}\right).$$

2.4.2 Tightness of Theorem 2.1.1 and Theorem 2.4.1

In this section, we show that the stretch guarantees in Theorem 2.1.1 and Theorem 2.4.1 are tight up to polylogarithmic factors.

Lemma 2.4.1 (Tightness of Theorem 2.1.1)**.** *For every large enough n, there exists an unweighted n vertex graph G, and a spectral sparsifier H of G such that $\widehat{H}$ has stretch $\widetilde{\Omega}(n^{2/3})$ w.r.t. G.*

Proof. As was shown by Spielman and Srivastava [30], one can create a sparsifier H of G (with high probability) by adding each edge e of G to H with probability $p_e = \min\{\epsilon^{-2} \cdot R^G_e \cdot \log n, 1\}$ (and weight $1/p_e$). This approach is known as *spectral sparsification using effective resistance sampling*. We will construct a graph G and argue that for a random graph H sampled according to the scheme above [30], the stretch of $\widehat{H}$ will (likely) be $\widetilde{\Omega}(n^{2/3})$.

For brevity, we will construct a graph with $n+2$ vertices and ignore rounding issues. The graph $G = (V, E)$ is constructed as follows. Let $N := \frac{n^{\frac{2}{3}}}{c}$ for $c := \log n$. We partition the set of vertices, V, into $V_0, V_1, \ldots, V_N, V_{N+1}$, where for each $i \in [1, N]$, we have $|V_i| = a = cn^{\frac{1}{3}}$, and $V_0 = \{u\}$, $V_{N+1} = \{v\}$ are singletons. For every $i \in [0, N]$, we connect all vertices in V_i to all vertices in V_{i+1}, and furthermore, we connect u and v by an edge called e. That is,

$$E = \left(\cup_{i=0}^{N} V_i \times V_{i+1}\right) \cup \{(u, v)\}.$$

See Figure 2.3 for illustration. Next, we calculate R^G_e, by observing the flow vector when one units of flow

is injected in v and is removed from u. Denote $R := R_e^G$. Then R units of flow is routed using edge e, while $(1-R)$ units of flow is routed using the rest of the graph. By symmetry, for each cut $V_i \times V_{i+1}$ the flow will spread equally among the edges. Farther, the potential of all the vertices in each set V_i is equal. Denote by P_i the potential of vertices in V_i. Thus $0 = P_0 < P_1 < \cdots < P_{N+1} = R$. For $i = 0$, each edge in $V_0 \times V_1$ carries $\frac{(1-R)}{a}$ flow, thus $P_1 - P_0 = \frac{(1-R)}{a}$. Similarly, $P_{N+1} - P_N = \frac{(1-R)}{a}$. On the other hand, for $i \in [1, N-1]$, each edge in $V_{i+1} \times V_i$ carries $\frac{(1-R)}{a^2}$ flow, thus $P_{i+1} - P_i = \frac{(1-R)}{a^2}$. We conclude

$$R = P_{N+1} - P_0 = \sum_{i=0}^{N} (P_{i+1} - P_i) = 2 \cdot \frac{(1-R)}{a} + \frac{(1-R)}{a^2} \cdot (N-1) = (1-R) \cdot \frac{2a + |N| - 1}{a^2}$$

Thus,

$$R_e^G = R = \frac{2a + |N| - 1}{a^2 - 2a - |N| + 1} = \frac{2cn^{1/3} + \frac{n^{2/3}}{c} - 1}{c^2 n^{2/3} - 2cn^{1/3} - \frac{n^{2/3}}{c} + 1} = \frac{1}{c^3}(1 + o(1)) = O(\frac{1}{\log^3 n}) .$$

Note that it is thus most likely that e will not belong to H (for large enough n). For a sampled graph H excluding e, we will have $d_{\widehat{H}}(u, v) \geq |N| = \widetilde{\Omega}(n^{2/3})$. From the other hand, as a graph H sampled in this manner is a spectral sparsifier with high probability, it implies the existence of a spectral sparsifier H of G with stretch $\widetilde{\Omega}(n^{2/3})$, as required. □

Lemma 2.4.2 (Tightness of Theorem 2.4.1)**.** *For every large enough m, there exists an unweighted graph G with m edges, and a spectral sparsifier H of G such that $\widehat{H}$ has stretch $\widetilde{\Omega}(\sqrt{m})$ w.r.t. G.*

Proof. Fix $n = (\frac{m}{2\log m})^{3/4}$. Note that the graph we constructed during the proof of Lemma 2.4.1 has $2a + (N-1)a^2 = 2cn^{1/3} + (\frac{n^{2/3}}{c} - 1) \cdot c^2 n^{2/3} < 2c \cdot n^{4/3} < m$ edges. We can complement it to exactly m edges by adding some isolated component. Following Lemma 2.4.1, this graph has a sparsifier H, such that $\widehat{H}$ has stretch $\widetilde{\Omega}(n^{2/3}) = \widetilde{\Omega}(\sqrt{m})$ w.r.t. G, as required.

□

Remark 2.4.1. *Cut sparsifiers are somewhat weaker version of spectral sparsifiers. Specifically, a weighted subgraph H of G is called a cut sparsifier if it preserves the size of all cuts (up to $1 \pm \varepsilon$ factor). A natural question is the following: given a cut sparsifier H of G, how good of a spanner is $\widehat{H}$?*
The answer is: very bad. Specifically, consider the hard instance constructed during the proof of Lemma 2.4.1. Construct the same graph G where we change the parameter N to equal $\Theta(\frac{n}{\log n})$ and a to $\Theta(\log n)$. There exist a cut sparsifier H of G excluding the edge $e = (u, v)$. In particular, $\widehat{H}$ will have stretch $\widetilde{\Omega}(n)$.

2.4.3 Stretch-Space trade-off

In this section, we first prove a result, which given an algorithm that uses $\widetilde{O}(n)$ space in the dynamic streaming setting, converts it to an algorithm that uses $\widetilde{O}(n^{1+\alpha})$ space and achieves a better stretch guarantee (see Theorem 2.4.2). Then, we apply this theorem to Corollary 2.1.1 and get a space-stretch trade off. Next, in Theorem 2.4.3 we prove a similar trade off in terms of number of edges.

Theorem 2.4.2. *Assume there is an algorithm, called* ALG*, that given a graph $G = (V, E)$ in a dynamic stream, with $|V| = n$, using $\widetilde{O}(n)$ space, outputs a spanner with stretch $\widetilde{O}(n^{\beta})$ for some constant $\beta \in (0, 1)$*

with failure probability n^{-c} for some constant c. Then, for any constant $\alpha \in (0,1)$, one can construct an algorithm that uses $\widetilde{O}(n^{1+\alpha})$ space and outputs a spanner with stretch $\widetilde{O}(n^{\beta(1-\alpha)})$ with failure probability $\tilde{O}(n^{(2+c)\alpha-c})$.

Proof. Let $\mathscr{P} \subset 2^{[n]}$ be a set of subsets of $[n]$ such that: (1) $|\mathscr{P}| = O(n^{2\alpha}\log n)$, (2) every $P \in \mathscr{P}$ is of size $|P| = O(n^{1-\alpha})$, and (3) for every $i, j \in [n]$ there is a set $P \in \mathscr{P}$ containing both i, j. Such a collection $\mathscr{P}$ can be constructed by a random sampling. Denote $V = \{v_1, \ldots, v_n\}$. For each $P \in \mathscr{P}$, set $A_P = \{v_i \mid i \in P\}$. For each $P \in \mathscr{P}$, we use ALG independently to construct a spanner H_P for $G[A_P]$ the induced graph on A_P. The final spanner will be their union $H = \cup_{P\in\mathscr{P}} H_P$.

The space (and also the number of edges in H) used by our algorithm is bounded by $\sum_{P\in\mathscr{P}} \widetilde{O}(|P|) = \widetilde{O}(n^{2\alpha} \cdot n^{1-\alpha}) = \widetilde{O}(n^{1+\alpha})$. From the other hand, for every $v_i, v_j \in V$ such that $i, j \in P$, it holds that

$$d_H(v_i, v_j) \leq d_{H_P}(v_i, v_j) \leq \widetilde{O}(|P|^{\beta}) \leq \widetilde{O}(n^{\beta(1-\alpha)}) .$$

By union bound, the failure probability is bounded by $\tilde{O}(n^{2\alpha}) \cdot O(n^{-c(1-\alpha)}) = \tilde{O}(n^{(2+c)\alpha-c})$.

□

Combining Corollary 2.1.1 with Theorem 2.4.2, we conclude:

Corollary 2.1.2. *Consider an n-vertex unweighted graph G, the edges of which arrive in a dynamic stream. For every parameter $\alpha \in (0,1)$, there is an algorithm using $\widetilde{O}(n^{1+\alpha})$ space, constructs a spanner with stretch $\widetilde{O}(n^{\frac{2}{3}(1-\alpha)})$ with high probability.*

Remark 2.4.2. *We can reduce the number of edges in the spanner returned to $O(n)$, by incurring additional $O(\log n)$ factor to the stretch. This is done by computing additional spanner upon the one returned by Theorem 2.1.2.*

Following the approach in Theorem 2.4.2, we can also use more space to reduce the stretch parameterized by the number of edges. Note that the Theorem 2.4.3 provides better result than Theorem 2.1.2 when $m \leq n^{\frac{4}{3}+\frac{2}{3}\alpha}$.

Theorem 2.4.3. *Consider an n-vertex unweighted graph G, the edges of which arrive in a dynamic stream. For every parameter $\alpha \in (0,1)$, there is an algorithm using $\widetilde{O}(n^{1+\alpha})$ space, constructs a spanner with stretch $\widetilde{O}(\sqrt{m} \cdot n^{-\alpha})$.*

Proof. Similarly to Theorem 2.4.2, our goal here is to partition the vertices into $\approx n^{2\alpha}$ sets of similar size. However, while in Theorem 2.4.2 we wanted to bound the number of vertices in each set, here we want to bound the edges in each set. As the edge set is unknown, we cannot use a fixed partition. Rather, in the preprocessing phase we will sample a partition that w.h.p. will be good w.r.t. arbitrary fixed edge set.

Fix $p = n^{-\alpha}$. With no regard to the rest of the algorithm, during the stream we will sample $\widetilde{O}(n^{1+\alpha}) = \widetilde{O}(\frac{n}{p})$ edges from the stream using sparse recovery (Lemma 2.2.2), and add them to our spanner $\widehat{H}$. If $m \leq np^{-1}$,

we will restore the entire graph G, and thus will have stretch 1. The rest of the analysis will be under the assumption that $m > np^{-1}$.

For every $i \in [1, \frac{8}{p^2} \ln n]$, sample a subset A_i by adding each vertex with probability p. Consider a single subset A_i sampled in this manner, and denote $G_i = G[A_i]$ the graph it induces. We will compute a sparsifier H_i for G_i. Our final spanner will be $\widehat{H} = \cup_i \widehat{H}_i$ a union of the unweighted versions of all the sparsifiers (in addition to the random edges sampled above). The space we used for the algorithm is $\sum_i \widetilde{O}(|A_i|)$. Note that with high probability, by Chernoff inequality $\sum_i \widetilde{O}(|A_i|) = \widetilde{O}(n^{2\alpha} \cdot n^{1-\alpha}) = \widetilde{O}(n^{1+\alpha})$.

Next we bound the stretch. Consider a pair of vertices $(u,v) \in E$. Denote by ψ_i the event that both u, v belong to A_i. Note that $\mathbb{P}[\psi_i] = p^2$. Denote by $m_i = \left|\binom{A_i}{2} \cap E\right|$ the number of edges in G_i. Set

$$\mu_i = \mathbb{E}\left[m_i \mid \psi_i\right] \leq 1 + p \cdot \left(\deg_G(v) + \deg_G(u)\right) + mp^2 < 1 + 2np + mp^2 < 4mp^2 \,,$$

to be the expected number of edges in G_i provided that $u, v \in A$. The first inequality follows as (1) $(u,v) \in G_i$, (2) every edge incident on u, v belongs to G_i with probability p, and (3) every other edge belongs to G_i with probability p^2. In the final inequality we used the assumption $n < mp$. Denote by ϕ_i the event that $m_i \leq 8mp^2$. By Markov we have

$$\mathbb{P}\left[\psi_i \wedge \phi_i\right] = \mathbb{P}\left[\psi_i\right] \cdot \mathbb{P}\left[\phi_i \mid \psi_i\right] \geq \frac{1}{2} p^2 \,.$$

As $\{\psi_i \wedge \phi_i\}_i$ are independent, we have that the probability that none of them occur is bounded by

$$\mathbb{P}\left[\bigwedge_i \left(\overline{\psi_i \wedge \phi_i}\right)\right] \leq (1 - \frac{1}{2}p^2)^{\frac{8}{p^2} \ln n} < e^{-\frac{1}{2}p^2 \cdot \frac{8}{p^2} \ln n} = n^{-4} \,.$$

Note that if both ψ_i, ϕ_i occurred, and H_i is an $1 \pm \varepsilon$ sparsifier of G_i, by Theorem 2.1.1 we will have that

$$d_{\widehat{H}}(u,v) \leq d_{\widehat{H}_i}(u,v) \leq \widetilde{O}(\sqrt{m_i}) = \widetilde{O}(\sqrt{m} \cdot p) = \widetilde{O}(\sqrt{m} \cdot n^{-\alpha})$$

By union bound, the probability that for every $(u,v) \in E$, there is some i such that $\psi_i \wedge \phi_i$ occurred is at least $1 - n^{-2}$. The probability that every G_i is a spectral sparsifier is $1 - n^{-\Omega(1)}$. The theorem follows by union bound.

□

2.5 Simultaneous Communication Model

In Section 2.4, we considered streaming model and proved results for the setting when one pass over the stream was allowed. The remaining question is as follows: using small number of communication rounds (but more than 1), can we improve the stretch of a spanner constructed in the simultaneous communication model? A partial answer is given in the following subsections.

First, in Section 2.5.1 we present a single filtering algorithm that provides two different trade-offs between stretch and number of communication rounds (see Algorithm 3 and Theorem 2.1.2). Basically, the

algorithm receives a parameter $t > 1$, in each communication round, an unweighted version of a sparsifier is added to the spanner. Then, locally in each vertex, all the edges that already have a small stretch in the current spanner are deleted (stop being considered), and another round of communication begins.

In Theorem 2.1.2 we present two arguments. The first argument is based on effective resistance filtering, which results in a spanner with $\tilde{O}(n^{\frac{g+1}{2g+1}})$ stretch in g communication rounds. The second argument, which is based on low-diameter decomposition, results in a spanner with $\tilde{O}\left(n^{\frac{2}{g}}\right)$ stretch in g communication rounds. The latter approach outputs a spanner with smaller stretch compared to the former algorithm for $g \geq 4$.

Finally, in Subsection 2.5.2 we generalize our results to the case where each player is allowed $\tilde{O}(n^{\alpha})$ communication per round for some $\alpha \in (0,1)$. In that section, we prove two results: (1) in Theorem 2.1.3 we give a space (communication per player) stretch trade off for one round of communication (2) in Theorem 2.1.4 we give a similar trade off for more than one round of communication.

2.5.1 The filtering algorithm

The algorithm will receive a stretch parameter t. During the execution of the algorithm, we will hold in each step a spanner $\widehat{H}$, and a subset of unsatisfied edges. As the algorithm proceeds, the spanner will grow, while the number of unsatisfied edges will decrease. Initially, we start with an empty spanner $\widehat{H}$, and the set of unsatisfied edges $E_0 = E$ is the entire edge set. In general, at round i, we hold a set E_i of edges yet unsatisfied. We construct a spectral sparsifier H_i for the graph $G_i = (V, E_i)$ over thus edges. $\widehat{H}_i$, the unweighted version of H_i is added to the spanner $\widehat{H}$. E_{i+1} is defined to be all the edges $(u,v) \in E_i$, for which the distance in $\widehat{H}$ is greater than t, that is $d_{\widehat{H}}(u,v) > t$. Note that as the sparsifier H_i, and hence the spanner $\widehat{H}$ is known to all, each vertex locally can compute which of its edges belong to E_{i+1}.

In addition, the algorithm will receive as an input parameter g to bound the number of communication rounds. We denote by E_{g+1} the set of unsatisfied edge by the end of the algorithm. That is edges from $(u,v) \in E$ for which $d_{\widehat{H}}(u,v) > t$. Note that during the execution of the algorithm, $E_{g+1} \subseteq E_g \subseteq E_{g-1} \subseteq \cdots \subseteq E_1 = E$. Finally, if $E_{g+1} = \emptyset$, it will directly imply that $\widehat{H}$ is a t-spanner of G. See Algorithm 3 for illustration.

Algorithm 3 `Spanners Using Filtering`$(G = (V,E), t, g)$

1: **procedure** SPANNERSUSINGFILTERING$(G = (V,E), t, g)$
▷ g is the number of rounds and t is the stretch parameter
2: $\epsilon \leftarrow \frac{1}{18}$
3: $\widehat{H} \leftarrow \emptyset$ ▷ $\widehat{H}$ will be the output spanner
4: **for** $i = 1$ to g **do**
5: $E_i \leftarrow \{e = (u,v) \in G_{i-1}$ such that $d_{\widehat{H}}(u,v) > t\}$
6: $G_i \leftarrow (V, E_i)$
7: Let H_i be a $(1 \pm \epsilon)$-spectral sparsifier of graph G_i
8: $\widehat{H} \leftarrow \widehat{H} \cup \widehat{H}_i$ ▷ $\widehat{H}_i$ is the unweighted version of H_i
9: **return** $\widehat{H}$ ▷ t-spanner of G with $\tilde{O}(n \cdot g)$ edges

Below, we state the theorem, which proves the round complexity and correctness of Algorithm 3.

Theorem 2.1.2. *For any integer $g \geq 1$, there is an algorithm (see Algorithm 3) that in g rounds of communication outputs a spanner with stretch* $\min\left\{\tilde{O}(n^{\frac{g+1}{2g+1}}),\ (12+o(1))\cdot n^{2/g}\cdot\log n\right\}$.

Proof. For $g=1$, the theorem holds due to Theorem 2.1.1, thus we will assume that $g \geq 2$. We prove each of the two upper-bounds on stretch separately. We prove the first bound using an effective resistance based argument. The latter upper-bound is proven using an argument based on filtering low-diameter clusters.

Effective resistance argument: We execute Algorithm 3 with parameter g, and $t = \tilde{O}(n^{\frac{g+1}{2g+1}})$. Consider an edge $e=(u,v) \in E_1$. If $e \in E_2$, then it follows from Corollary 2.4.1 that $R^{H_1}_{u,v} = \tilde{\Omega}\left(\frac{t^3}{n^2}\right)$. Set $a_1 = \tilde{\Omega}\left(\frac{t^3}{n^2}\right)$. Then

$$|E_2| \leq \frac{1}{a_1}\sum_{e\in E_1} R^{H_1}_e \leq \frac{1+\varepsilon}{a_1}\sum_{e\in E_i} R^{G_1}_e \leq \frac{1+\varepsilon}{a_1}\cdot(n-1) \leq \tilde{\Omega}\left(\frac{n^3}{t^3}\right), \tag{2.7}$$

where the first inequality follows as $a_1 \leq R^{H_1}_e$ for $e \in E_2$, the second inequality is by Fact 2.2.2, and the third inequity follows by Fact 2.2.3, as G_{i-1} is unweighted. In general, for $i \geq 2$, we argue by induction that $|E_i| = \tilde{O}\left(\frac{n^{i+1}}{t^{2i-1}}\right)$. Indeed, consider an edge $e \in E_{i+1}$. Using the induction hypothesis, it follows from Corollary 2.4.2 that

$$R^{H_i}_{u,v} = \tilde{\Omega}\left(\frac{t^2}{|E_i|}\right) = \tilde{\Omega}\left(\frac{t^{2(i+1)-1}}{n^{i+1}}\right)$$

Set $a_i = \tilde{\Omega}\left(\frac{t^{2(i+1)-1}}{n^{i+1}}\right)$. Using the same arguments as in (2.7), we get

$$|E_{i+1}| \leq \frac{1}{a_i}\sum_{e\in E_i} R^{H_i}_e \leq \frac{1+\varepsilon}{a_i}\sum_{e\in E_i} R^{G_i}_e \leq \frac{1+\varepsilon}{a_i}\cdot(n-1) \leq \tilde{O}\left(\frac{n^{(i+1)+1}}{t^{2(i+1)-1}}\right).$$

Finally, for every $e \in E_g$, following Theorem 2.4.1, it holds that

$$d_{\widehat{H}}(u,v) \leq d_{\widehat{H}_g}(u,v) \leq \tilde{O}\left(\sqrt{E_g}\right) = \tilde{\Omega}\left(\sqrt{\frac{n^{g+1}}{t^{2g-1}}}\right) \leq t,$$

where the last inequality holds for $t = \tilde{\Omega}(n^{\frac{g+1}{2g+1}})$. We conclude that $E_{g+1} = \emptyset$. The theorem follows.

Low diameter decomposition argument: Fix $\phi = \frac{1}{3}n^{-2/g}$. We will execute Algorithm 3 with parameter g and $t = \frac{4+o(1)}{\phi}\cdot\ln n$. We argue that for every $i \in [2, g+1]$, $|E_{i+1}| \leq 3\phi|E_i|$. As $|E_1| < n^2$, it will follow that $E_{g+1} = \emptyset$, as required.

Consider the unweighted graph G_i, and the sparsifier H_i we computed for it. We will cluster G_i based on cut sizes in H_i. The clustering procedure is iterative, where in phase j we holds an induced subgraph $H_{i,j}$ of H_i, create a cluster C_j, remove it from the graph $H_{i,j}$ to obtain an induced subgraph $H_{i,j+1}$, and continue. The procedure stops once all the vertices are clustered. Specifically, in phase j, we

pick an arbitrary unclustered center vertex $v_j \in H_{i,j}$, and create a cluster by growing a ball around v_j. Set $B_r = B_{\hat{H}_{i,j}}(v_j, r)$ to be the radius r ball around v_j in the unweighted version of $H_{i,j}$. That is $B_{r+1} = B_r \cup N(B_r)$, where $N(B_r)$ are the neighbors of B_r in $H_{i,j}$. Let r_j be the minimal index r such that

$$\partial_{H_{i,j}}(B_r) < \phi \cdot \mathrm{Vol}_{H_{i,j}}(B_r) \,. \tag{2.8}$$

Here $\partial_{H_{i,j}}(B_r)$ denotes the total weight of the outgoing edges from B_r, while $\mathrm{Vol}_{\hat{H}_{i,j}}(B_r) = \sum_{u \in B_r} \deg_{\hat{H}_{i,j}}(u)$ denotes the sum of the weighted degrees of all the vertices in B_r. Note that while B_r is defined w.r.t. an unweighted graph $\widehat{H}_{i,j}$, $\partial_{H_{i,j}}$ and $\mathrm{Vol}_{H_{i,j}}$ are defined w.r.t. the weighted sparsifier. For every r, it holds that $\mathrm{Vol}_{\hat{H}_{i,j}}(B_{r+1}) \geq \mathrm{Vol}_{\hat{H}_{i,j}}(B_r) + \partial_{\hat{H}_{i,j}}(B_r)$. We argue that $r_j \leq 2(1+\epsilon) \cdot \binom{n}{2}$. If v_j is isolated in $H_{i,j}$, then (2.8) holds for $r = 0$ and we are done. Else, as the minimal weight of an edge in a sparsifier is 1,[5] it holds that $\mathrm{Vol}_{H_{i,j}}(B_0) = \deg_{H_{i,j}}(v_j) \geq 1$. We conclude that for r_j, the minimal index for which (2.8) holds, we have that

$$\mathrm{Vol}_{H_{i,j}}(B_{r_j}) \geq (1+\phi)\mathrm{Vol}_{H_{i,j}}(B_{r-1}) \geq \cdots \geq (1+\phi)^{r_j}\mathrm{Vol}_{H_{i,j}}(B_0) \geq (1+\phi)^{r_j} \,,$$

On the other hand, as H_i is a $(1+\varepsilon)$ spectral sparsifier of an unweighted graph G_i, we have

$$\mathrm{Vol}_{H_{i,j}}(B_{r_j}) \leq \mathrm{Vol}_{H_{i,j}}(H_{i,j}) \leq 2(1+\epsilon) \cdot |E| \leq 2(1+\epsilon) \cdot \binom{n}{2} .$$

Therefore, it must holds that $(1+\phi)^{r_j} \leq 2(1+\epsilon)\binom{n}{2}$, which implies

$$r_j \leq \frac{\ln((1+\varepsilon)n^2)}{\ln(1+\phi)} = \frac{2+o(1)}{\phi} \cdot \ln n \,.$$

We set $C_j = B_{r_j}$ and continue to construct C_{j+1}. Overall, we found a partition of the vertex set V into clusters $C_1, C_2, \ldots$ such that each cluster satisfies (2.8), and has (unweighted) diameter at most $\frac{4+o(1)}{\phi} \cdot \ln n = t$. In particular, for every edge $e = (u,v) \in E_i$, if u, v are clustered to the same C_i, then the distance between them in $\hat{H}$ will be bounded by t. Thus E_{i+1} will be a subset $\partial_{H_i}(C_1, C_2, \ldots)$, the set of inter-cluster edges. It holds that

$$\partial_{H_i}(C_1, C_2, \ldots) = \sum_{j \geq 1} \partial_{H_{i,j}}(C_j) \leq \phi \cdot \sum_{j \geq 1} \mathrm{Vol}_{H_{i,j}}(C_j) \leq \phi \cdot \mathrm{Vol}_{H_i}(V) \,, \tag{2.9}$$

where the first inequality holds as each edge counted exactly once. For example the edge $(u,v) \in E(C_a, C_b)$, where $a < b$ counted only at $\partial_{\hat{H}_{i,a}}(C_a)$. Hence,

$$\begin{aligned}
|E_{i+1}| \leq \partial_{G_i}(C_1, C_2, \ldots) &\leq (1+\epsilon)\partial_{H_i}(C_1, C_2, \ldots) && \text{By Fact 2.2.1} \\
&\leq (1+\epsilon)\phi \cdot \mathrm{Vol}_{H_i}(V) && \text{By (2.9)} \\
&\leq \frac{\phi(1+\epsilon)}{1-\epsilon} \cdot \mathrm{Vol}_{G_i}(V) && \text{By Fact 2.2.1} \\
&= \phi \cdot \left(1 + \frac{2\epsilon}{1-\epsilon}\right) \cdot 2|E_i| < 3\phi \cdot |E_i| \,.
\end{aligned}$$

[5] Since we are producing spectral sparsifiers by effective resistance sampling method using corresponding sketches, each edge e is reweighted by $\frac{1}{p_e}$ where p_e is the probability that edge e is sampled, and hence the weights are at least 1.

□

Remark 2.5.1. *Note that in fact for the low diameter decomposition argument, it is enough to use in Algorithm 3 cut sparsifiers rather than spectral sparsifiers.*

2.5.2 Stretch-Communication trade-off

We note that if more communication per round is allowed, then we can obtain the following.

Theorem 2.1.3. *There is an algorithm that in* 1 *round of communication, where each player communicates* $\tilde{O}(n^{\alpha})$ *bits, outputs a spanner with stretch*

$$\min\left\{\widetilde{O}(n^{(1-\alpha)\frac{2}{3}}),\ \widetilde{O}\left(\sqrt{m}\cdot n^{-\alpha}\right)\right\}.$$

Proof. We prove the stretch bounds, one by one.

Proving $\widetilde{O}(n^{(1-\alpha)\frac{2}{3}})$: Basically, the claim follows by Corollary 2.1.1. More specifically, we work on graphs induces on $O(n^{1-\alpha})$ sized set of vertices. For each such subgraph, we can construct sparsifiers using $O(\text{polylog}(n))$ sized sketches communicated by each vertex involved. Since each vertex is involved in $\widetilde{O}(n^{\alpha})$ subgraphs, then communication per vertex is $\widetilde{O}(n^{\alpha})$. And by Corollary 2.4.1, the stretch is $\widetilde{O}(n^{(1-\alpha)\frac{2}{3}})$.

Proving $\widetilde{O}\left(\sqrt{m}\cdot n^{-\alpha}\right)$: First, the reader should note that we cannot directly use Corollary 2.4.2 for this part. The reason is that during the proof of Theorem 2.4.3, for the special case where $m \le n^{1+\alpha}$, we simply used a sparse recovery procedure to recover the entire graph. However, as the graph G might contain a dense subgraph, sparse recovery is impossible in the simultaneous communication model. Instead, we use a procedure, called *peeling low degree vertices*, where using $\widetilde{O}(n^{\alpha})$ bits of information per vertex, we can partition the vertices into two sets, V_1 and V_2, where all edges incident on V_1 are recovered and minimum degree in $G[V_2]$ is at least $n^{-\alpha}$. We present this procedure in Algorithm 4 and its guarantees are proved in Claim 2.5.1 below.

Lemma 2.5.1 (Peeling low-degree vertices)**.** *In a simultaneous communication model, where communication per player is* $\tilde{O}(s)$*, there is an algorithm that each vertex can locally run and output a partition of the vertices into* V_1, V_2 *such that:*

1. *All the incident edges of* V_1 *are recovered.*
2. *The min-degree in the induce graph* $G[V_2]$ *is at least* s*.*

Furthermore, the partitions output by all vertices are identical, due to the presence of shared randomness.

Proof. First, we argue that using s-sparse recovery procedure on the neighborhood of vertices, one can find a set $V_1 \subseteq V$ such that all the vertices in $V \setminus V_1$ have degree more than s. This is done in the following

way: each vertex prepares an s-sparse recovery sketch for its neighborhood, and in the first round of communication writes its sketch alongside its degree on the board. Then, each vertex runs Algorithm 4 locally. Note that the output is identical in all vertices since they have access to shared randomness.

Now, we argue the correctness of Algorithm 4. First, we let RECOVER be a s-sparse recovery algorithm. More specifically, the following fact holds.

Fact 2.5.1. *For any integer s, given S, a $\widetilde{O}(s)$-bit sized linear s-sparse recovery sketch of a vector $\vec{b}$, such that* $\text{SUPPORT}(\vec{b}) \leq s$*, algorithm* $\text{RECOVER}(S)$ *outputs the non-zero elements of $\vec{b}$, with high probability.*

Consider the execution of Algorithm 4. If in the beginning there does not exist a low-degree vertex, we are done. Otherwise, there exists a vertex u with degree $\leq s$. Now, when we call $\text{RECOVER}(S_u)$ it is guaranteed that the support of the vector is bounded by s (see line 4 of Algorithm 4). In that case, $\text{RECOVER}(S_u)$ succeeds with high probability. Note that in case of success, the output of $\text{RECOVER}(S_u)$ is deterministic- that is depend only the graph and not on the random coins. Then, we delete vertex u alongside its incident edges. The sketches for the rest of the graph can be updated accordingly, since the sketches are linear. Thus, we can use the updated sketches in the next round to recover the neighborhood of another low-degree vertex (in the updated graph), without encountering dependency issues (as the series of events we should succeed upon is predetermined). We repeat this procedure until no vertex with degree $\leq s$ remains. Furthermore, we call RECOVER at most n times per vertex (since we can delete at most n vertices), in total, using union bound, the algorithm succeeds with high probability. [6]

Algorithm 4 Low-Degree Peeling($\{S_u\}_{u\in V}, s$)

1: **procedure** LOWDEGREEPEELING($\{S_u\}_{u\in V}, s$)
▷ linear s-sparse recovery sketches (denoted by S_u for each vertex u)
2: $V_1 \leftarrow \emptyset$
3: $V_2 \leftarrow V$
4: **while** $\exists$ a vertex u with degree $\leq s$ **do**
5: $u \leftarrow$ a vertex with degree $\leq s$ ▷ Using a universal ordering, and degrees in $G[V_2]$
6: $E_u \leftarrow \text{RECOVER}(S_u)$ ▷ See Fact 2.5.1
7: Remove E_u from the sketches and update degrees. ▷ Sketches are linear
8: $V_1 \leftarrow V_1 \cup \{u\}$.
9: $V_2 \leftarrow V_2 \setminus \{u\}$.
10: **return** (V_1, V_2)
▷ A partition of vertices into two sets, V_1 and V_2, with the guarantees mentioned in Claim 2.5.1

□

We use Algorithm 4 with $s = n^{\alpha}$. In the same time, we use the algorithm from Theorem 2.4.3. That is, partition the vertices into $\widetilde{O}(n^{2\alpha})$ sets such that each vertex belong to each set with probability $n^{-\alpha}$. Than compute a sparsifier H for each set and take their union. It follows that the total required communication

[6] A similar argument is also given in [90].

is $\widetilde{O}(n^{\alpha})$ per vertex. Note that the algorithm of Theorem 2.4.3 is linear. Hence after using Claim 2.5.1, we can add all the edges incident on V_1 to the spanner, and update the algorithm from Theorem 2.4.3 accordingly. That is we will use it only on $G[V_2]$.

Note that we have $|E(G[V_2])| \geq |V_2| \cdot n^{\alpha}$, and consequently we can use the argument in the proof of Theorem 2.4.3. In total from one hand we will obtain stretch 1 on edges incident to V_1, and from the other hand, for edges inside $G[V_2]$ we will have stretch of $\widetilde{O}(\sqrt{|E(G[V_2])|} \cdot n^{\alpha}) \leq \widetilde{O}(\sqrt{m} \cdot n^{\alpha})$. □

Theorem 2.1.4. *For any integer $g \geq 1$, there is an algorithm that in g rounds of communication, where each player communicates $\tilde{O}(n^{\alpha})$ bits, outputs a spanner with stretch*

$$\min\left\{(12+o(1)) \cdot n^{(1-\alpha)\cdot\frac{2}{g}} \cdot \log n \,,\; \widetilde{O}\left(n^{\frac{(g+1)(1-\alpha)}{2g+1}}\right)\right\} .$$

Proof. We use the same set of subsets of vertices as in Theorem 2.4.2, i.e., let $\mathscr{P} \subset 2^{[n]}$ be a set of subsets of $[n]$ such that: (1) $|\mathscr{P}| = O(n^{2\alpha}\log n)$, (2) every $P \in \mathscr{P}$ is of size $|P| = O(n^{1-\alpha})$, and (3) for every $i, j \in [n]$ there is a set $P \in \mathscr{P}$ containing both i, j. Such a collection $\mathscr{P}$ can be constructed by a random sampling. Denote $V = \{v_1, \ldots, v_n\}$. For each $P \in \mathscr{P}$, set $A_P = \{v_i \mid i \in P\}$. For each $P \in \mathscr{P}$, we use Algorithm 3 independently on each subgraph. Then, using Theorem 2.1.2 on each subgraph, since the size of each subgraph is $O(n^{1-\alpha})$ and since for each edge we have a subgraph that this edge is present, the claim holds. □

3 Kernel Density Estimation through Density Constrained Near Neighbor Search

This chapter is based on a joint work with Moses Charikar, Michael Kapralov and Paris Siminelakis. It has been accepted to the 61st IEEE Annual Symposium on Foundations of Computer Science [91, FOCS]

3.1 Introduction

Kernel density estimation is a fundamental problem with numerous applications in machine learning, statistics and data analysis [92, 93, 94, 95, 96, 97, 98]. Formally, the Kernel Density Estimation (KDE) problem is: preprocess a dataset P of n points $\mathbf{p}_1, \ldots, \mathbf{p}_n \in \mathbb{R}^d$ into a small space data structure that allows one to quickly approximate, given a query $\mathbf{q} \in \mathbb{R}^d$, the quantity

$$K(P, \mathbf{q}) := \frac{1}{|P|} \sum_{\mathbf{p} \in P} K(\mathbf{p}, \mathbf{q}). \tag{3.1}$$

where $K(\mathbf{p}, \mathbf{q})$ is the kernel function. The Gaussian kernel

$$K(\mathbf{p}, \mathbf{q}) := \exp(-||\mathbf{p} - \mathbf{q}||_2^2/2)$$

is a prominent example, although many other kernels (e.g., Laplace, exponential, polynomial etc) are the method of choice in many applications [99, 100].

In the rest of the paper, we use the notation μ^* defined as $\mu^* := K(P, \mathbf{q})$, and μ is a quantity that satisfies $\mu^* \leq \mu \leq 4\mu^*$. Moreover, in the statement of the main results, we assume that a constant factor lower bound to the actual kernel density, μ^*, is known. In general, if we only know that $\mu^* \geq \tau$ for some τ, then the μ^* terms in the space should be replaced by τ (similar to prior results in the literature). However, the query time can always be stated in terms of μ^*.

The kernel density estimation problem has received a lot of attention over the years, with very strong results available for low dimensional datasets. For example, the celebrated fast multipole method [101] and the related Fast Gauss Transform can be used to obtain efficient data structure for KDE (and in fact solves the more general problem of multiplying by a kernel matrix). However, this approach suffers from an exponential dependence on the dimension of the input data points, a deficiency that it shares with

other tree-based methods [102, 103, 104, 105, 106]. A recent line of work [8, 9, 107, 108] designed sublinear query algorithms for kernel density estimation in high dimensions using variants of the Locality Sensitive Hashing [8] framework of Indyk and Motwani [58].

Most of these works constructed estimators based on locality sensitive hashing, and then bounded the variance of these estimators to show that a small number of repetitions suffices for a good estimate. Bounding the variance of LSH-based estimators is nontrivial due to correlations inherent in sampling processes based on LSH, and the actual variance turns out to be nontrivially high.

In this work we take a different approach to implementing importance sampling for KDE using LSH-based near neighbor search techniques. At a high level, our approach consists of first performing independent sampling on the dataset, and then using using LSH-based near neighbor search primitives to extract relevant data points from this sample[1]. The key observation is that the sampled dataset in the KDE problem has nice geometric structure: the number of data points around a given query cannot grow too fast as a function of distance and the actual KDE value μ (we refer to these constraints as density constraints – see Section 3.2 for more details). The fact that our approach departs from the idea of constructing unbiased estimators of KDE directly from LSH buckets turns out to have two benefits: first, we immediately get a simple algorithm that uses classical LSH-based near neighbor search primitives (Euclidean LSH of Andoni and Indyk [10]) to improve on or essentially matches all prior work on kernel density estimation for radial kernels. The result is formally stated as Theorem 3.1.1 for the Gaussian kernel below, and its rather compact analysis in a more general form that extends to other kernels is presented in Section 3.4. The second benefit of our approach is that it distills a clean near neighbor search problem, which we think of as near neighbor search under density constraints, and improved algorithms for that problem immediately yield improvements for the KDE problem itself. This clean separation allows us to use the recent exciting data-depending techniques pioneered by [11, 12, 13] in our setting. It turns out that while it seems plausible that data-dependent techniques can improve performance in our setting, actually designing and analyzing a data-dependent algorithm for density constrained near neighbor search is quite nontrivial. The key difficulty here lies in the fact that one needs to design tools for tracking the evolution of the density of the dataset around a given query through a sequence of recursive partitioning steps (such evolution turns out to be quite involved, and in particular governed by a solution to an integral equation involving the log density of the kernel and properties of Spherical LSH). The design of such tools is our main technical contribution and is presented in Section 3.5. The final result for the Gaussian kernel is given below as Theorem 3.1.2, and extensions to other kernels are presented in Section 3.5.

3.1.1 Our results

We instantiate our results for the Gaussian kernel as an illustration, and then discuss extensions to more general settings. We assume that $\mu^* = n^{-\Theta(1)}$, since this is the interesting regime for this problem. For $\mu^* = n^{-\omega(1)}$ under the Orthogonal Vectors Conjecture (e.g. [109]), the problem cannot be solved faster than $n^{1-o(1)}$ using space $n^{2-o(1)}$ [9], and for larger values $\mu^* = n^{-o(1)}$ random sampling solves the problem in $n^{o(1)}/\epsilon^2$ time and space.

[1]The approach of [107] also used near neighbor search techniques, but was only using c-ANN primitives as a black box, which turns out to be constraining – this only leads to strong results for slowly varying kernels (i.e., polynomial kernels). Our data-independent result recovers the results of [107], up to a $\mu^{-o(1)}$ loss, as a special case.

Data-Independent LSH Our first result uses data-independent LSH of Andoni-Indyk [10] to improve upon the previously best known result [8] and follow up works that required query time $\widetilde{O}(\mu^{-0.5-o(1)}/\epsilon^2)$ if only polynomial space in $1/\mu$ is available.

Theorem 3.1.1. *Given a kernel $K(\mathbf{p},\mathbf{q}) := e^{-a||\mathbf{p}-\mathbf{q}||_2^2}$ for any $a > 0$, $\epsilon = \Omega\left(\frac{1}{\text{polylog}n}\right)$, $\mu^* = n^{-\Theta(1)}$ and a data set of points P, there exists an algorithm for preprocessing and an algorithm for query procedure such that after receiving query $\mathbf{q}$ one can approximate $\mu^* := K(P,\mathbf{q})$ (see Definition 3.4.1) up to $(1\pm\epsilon)$ multiplicative factor, in time $\widetilde{O}\left(\epsilon^{-2}\left(\frac{1}{\mu^*}\right)^{0.25+o(1)}\right)$, and the space consumption of the data structure is*

$$\min\left\{\epsilon^{-2}n\left(\frac{1}{\mu^*}\right)^{0.25+o(1)}, \epsilon^{-2}\left(\frac{1}{\mu^*}\right)^{1+o(1)}\right\}.$$

Remark 3.1.1. *In Theorem 3.1.1 (and similar theorems in the rest of the paper), we assumed that $\epsilon = \Omega\left(\frac{1}{\text{polylog}n}\right)$ and $\mu^* = n^{-\Theta(1)}$, so that we can assume $d = \widetilde{O}(1)$ (and ignore the dependencies on dimension in the statements). The reason (for $d = \widetilde{O}(1)$) is that in this case the contribution of far points (points at distance $\Omega(\log n)$) is negligible and for close points, we can use Johnson-Lindenstrauss (JL) lemma to reduce the dimension to $O(\text{polylog}n)$, without distorting the kernel value by a more than $1\pm o(1)$ multiplicative factor. If we remove these assumptions, we need to multiply the query-time and space bounds by dimension d.*

This theorem is stated and proved as Theorem 3.4.1 in Section 3.4. To get a sense of the improvement, the result of [8] exhibited query time that is roughly a square root of the query time of uniform random sampling. Our result uses the same LSH family as in [8] but achieves query time that is itself roughly the square root of that of [8]!

Data-Dependent LSH Our main technical contribution is a collection of techniques for using data dependent hashing introduced by [11, 12, 13] in the context of kernel density estimation. Unlike these works, however, who had no assumptions on the input data set, we show how to obtain refined bounds on the efficiency of near neighbor search under density constraints imposed by assumptions on KDE value as a function of the kernel. This turns out to be significantly more challenging: while in approximate near neighbor search, as in [13], it essentially suffices to track the size of the dataset in recursive iterations of locality sensitive hashing and partitioning into spheres, in the case of density constrained range search problems arising from KDE one must keep track of the distribution of points across different distance scales in the hash buckets, i.e. track evolution of functions as opposed to numbers. This leads to a natural linear programming relaxation that bounds the performance of our algorithm that forms the core of our analysis[2]. Our ultimate result for the Gaussian kernel is:

Theorem 3.1.2. *For Gaussian kernel K, any data set of points P and any $\epsilon = \Omega\left(\frac{1}{\text{polylog}n}\right)$, $\mu^* = n^{-\Theta(1)}$, using Algorithm 5 for preprocessing and Algorithm 6 for the query procedure, one can approximate $\mu^* := K(P,\mathbf{q})$ (see Definition 3.4.1) up to $(1\pm\epsilon)$ multiplicative factor, in time $\widetilde{O}(\mu^{-0.173-o(1)}/\epsilon^2)$. The space complexity of*

[2]The actual optimal evolution is described by an integral equation involving the log density of the kernel function and collision probabilities of LSH on the Euclidean sphere, but we do not make the limiting claim formal here since the ultimate integral equation appears to not have a closed form solution, and hence would not be useful for analysis purposes.

the algorithm is also bounded by

$$\min\left\{O(n\cdot\mu^{-(0.173+o(1))}/\epsilon^2), O\left(\mu^{-(1+c+o(1))}/\epsilon^2\right)\right\},$$

for $c = 10^{-3}$.[3]

The proof of Theorem 3.1.2 is given in Section 3.5.

Our techniques extend to other kernels – the extensions are presented in Section 3.5.

3.1.2 Related Work

For $d \gg 1$, KDE was studied extensively in the 2000's with the works of [102, 103, 104, 105, 106] that employed hierarchical space partitions (e.g. kd-trees, cover-trees) to obtain sub-linear query time for datasets with low intrinsic dimensionality [110]. Nevertheless, until recently [8], in the regime of $d = \Omega(\log n)$ and under worst case assumptions, the best known algorithm was simple random sampling that for constant $\delta > 0$ requires $O(\min\{1/\epsilon^2\mu, n\})$ evaluations of the kernel function to provably approximate the density at any query point q.

[8] revisited the problem and introduced a technique, called Hashing-Based-Estimators (HBE), to implement low-variance Importance Sampling (IS) efficiently for any query through Locality Sensitive Hashing (LSH). For the Gaussian $f(r) = e^{-r^2}$, Exponential $f(r) = e^{-r}$, and t-Student kernels $f(r) = (1+r^t)^{-1}$ the authors gave the first sub-linear algorithms that require $O(\min\{1/\epsilon^2\sqrt{\mu}, n\})$ kernel evaluations. Using ideas from Harmonic Analysis, the technique was later extended in [9], to apply to more general kernels resulting in the first data structures that require $O(\min\{1/\epsilon^2\sqrt{\mu}, n\})$ kernel evaluations to approximate the density for log-convex kernels $e^{\phi(\langle x,y\rangle)}$. Furthermore, under the Orthogonal Vectors Conjecture it was shown that there does not exist a data structure that solves the KDE problem under the Gaussian kernel in time $n^{1-o(1)}/\mu^{o(1)}$ and space $n^{2-o(1)}/\mu^{o(1)}$.

The work most closely related to ours is that of [107]. [107] introduced a technique, called Spherical Integration, that uses black-box calls to c-ANN data structures (constructed on sub-sampled versions of the data set) to sample points from "spherical annuli" (r, cr) around the query, for all annuli that had non-negligible contribution to the density of the query. For kernels with polynomial tails of degree t, their approach required $\widetilde{O}(c^{5t})$ calls to such data-structures (without counting the query time required for each such call) to estimate the density. Unfortunately, this approach turns out to be constraining due to its reliance on **black-box** c-ANN calls, and in particular only applies to polynomial kernels. Our techniques in this paper recover the result of [107] up to $\mu^{-o(1)}$ factors as a special case (see Section 3.4). Furthermore, the $\mu^{-o(1)}$ factor loss that we incur is only due to the fact that we are using the powerful Euclidean LSH family in order to achieve strong bounds for kernels that exhibit fast decay (e.g., Gaussian, exponential and others) using the same algorithm. For polynomial kernels the dependence on μ in our approach can be reduced to polylogarithmic in $1/\mu$ by using an easier hash family (e.g., the hash family of [111]; see [112, Chapter 10] for details).

[3] This c can be set to any small constant that one desires. For our setting of parameters $c = 10^{-3}$.

Scalable approaches to KDE and Applications Recent works [113, 108] also address scalability issues of the original approach of [8]. [113] designed a more efficient adaptive procedure that can be used along with Euclidean LSH [111] to solve KDE for a variety of power-exponential kernels, most prominently the Gaussian. Their algorithm is the first practical algorithm for Gaussian KDE with worst case guarantees that improve upon random sampling in high dimensions. Experiments in real-world data sets show [113] that the method of [8], yields practical improvements for many real world datasets. [108] introduced a way to sparsify hash tables and showed that in order to estimate densities $\mu^* \geq \tau \geq \frac{1}{n}$ one can reduce the space usage of the data structures [113] from $O(1/\tau^{3/2}\epsilon^2)$ to $O(1/\tau\epsilon^2)$. The authors also evaluated their approach on real world data for the Exponential $e^{-\|x-y\|_2}$ and Laplace $e^{-\|x-y\|_1}$ kernels showing improvements compared to [8] and uniform random sampling. A related approach of Locality Sensitive Samplers [114] has also been applied to obtain practical procedures in the contexts of Outlier detection [115], Gradient Estimation [116] and Clustering [117]. Finally, [118] uses similar ideas to address the problem of approximate range counting on the unit sphere.

Core-sets and Kernel sketching The problem of KDE is phrased in terms of guarantees for any single query $\mathbf{q} \in \mathbb{R}^d$. A related problem is that of Core-sets for kernels [119], where the goal is to find a (small) set $S \subset P$ such that the kernel density estimate on P is close to the one on S. After recent flurry of research efforts [120, 121] has resulted in near optimal [121] unweighted $|S| = O(\sqrt{d\log(1/\epsilon)}/\epsilon)$ and optimal [122] weighted core-sets $|S| = O(\sqrt{d}/\epsilon)$ for positive definite kernels. Somewhat related to this problem is the problem of oblivious sub-space embeddings for polynomial kernels [123, 124, 125, 126].

3.1.3 Outline

We start by giving a technical overview of the paper in Section 3.2. Preliminary definitions and results are presented in Section 3.3. In Section 3.4, we present our data-independent result for Gaussian KDE and state a general version of our result for other decreasing kernels. We present our data structure based on Data-Dependent LSH for Gaussian KDE in Section 3.5 and its analysis in Sections 3.6 (Query time), 3.7 (Valid execution path analysis), 3.8 (Linear Program analysis), and 3.9 (Primal-Dual solution).

3.2 Technical overview

In this section we give an overview of our results and the main ideas behind them. For simplicity we use the Gaussian kernel, even though both our results extend to more general settings. Thus, for the purposes of this overview our problem is: preprocess a dataset P of n points $\mathbf{p}_1, \ldots, \mathbf{p}_n \in \mathbb{R}^d$ into a small space data structure that allows fast KDE queries, i.e. can quickly approximate, given $\mathbf{q} \in \mathbb{R}^d$, the quantity

$$K(P, \mathbf{q}) := \frac{1}{|P|} \sum_{\mathbf{p} \in P} K(\mathbf{p}, \mathbf{q}), \tag{3.2}$$

where

$$K(\mathbf{p}, \mathbf{q}) := \exp(-||\mathbf{p} - \mathbf{q}||_2^2/2).$$

We present two schemes based on ideas from data independent and data dependent LSH schemes. Both schemes employ the strategy of first sampling the dataset at a sequence of geometric levels, and then using near neighbor search algorithms to retrieve all points at an appropriate distance from the query from the sample. The difference between the two approaches lies in the implementation and analysis of the near neighbor search primitive used for this retrieval. In what follows we first overview our approach to implementing importance sampling for KDE using near neighbor search primitives, and then instantiate this scheme with data-independent (Section 3.2.1) and data-dependent (Section 3.2.2) schemes.

3.2.1 Data-independent algorithm (Section 3.4)

We start by showing a new application of data-independent locality sensitive hashing to KDE that results in a simple scheme that provides the following result.

Theorem 3.2.1 (Informal version of Theorem 3.4.1)**.** *If $\mu^* := K(P,\mathbf{q})$, then there exists an algorithm that can approximate μ^* up to $(1\pm\epsilon)$ multiplicative factor, in time $\left(\frac{1}{\mu^*}\right)^{0.25+o(1)}$, using a data structure of size*

$$\min\left\{\epsilon^{-2}n\left(\frac{1}{\mu^*}\right)^{0.25+o(1)},\epsilon^{-2}\left(\frac{1}{\mu^*}\right)^{1+o(1)}\right\}.$$

We remark that the actual non-adaptive algorithm that we present in Section 3.4 is more general than the above and applies to a wide class of kernels. In particular, it simultaneously improves upon all prior work on radial kernels that exhibit fast tail decay (such as the exponential and the Gaussian kernels) [8] as well as matches the result of [107] on kernels with only inverse polynomial rate of decay up to $\mu^{-o(1)}$ factors.

We now outline the algorithm and the analysis. The main idea is simple: we note that in order to approximate the sum on the right hand side of (3.2), ideally we would like to do importance sampling, i.e. pick every point with probability proportional to its contribution to the KDE value. It is of course not immediate how to do this, since the contribution depends on the query, which we do not know at the preprocessing stage. However, we show that it is possible to simply prepare sampled versions of the input dataset using a fixed geometric sequence of sampling rates, and then use locality sensitive hashing to retrieve the points relevant to the given query from this sample efficiently. Below, we present an overview of our algorithm.

Geometric weight levels: Let $J := \lceil \log \frac{1}{\mu} \rceil$ and partition the points in the data set into J sets, such that the contribution of any point in the j'th set to the kernel density is $\approx 2^{-j}$. If $w_i := K(\mathbf{p}_i,\mathbf{q})$, then we define (see Definition 3.4.2) level sets

$$L_j := \left\{\mathbf{p}_i \in P : w_i \approx 2^{-j}\right\}.$$

The kernel density can be expressed in terms of the level sets as

$$K(P,\mathbf{q}) \approx \frac{1}{n}\sum_{j=1}^{J} |L_j|\cdot 2^{-j},$$

which implies size upper bounds for L_j, namely:

$$|L_j| \lesssim 2^j n\mu. \tag{3.3}$$

This means that for every query $\mathbf{q}$ such that the KDE value at $\mathbf{q}$ equals μ to within constant factors one can place an upper bound of $2^j n\mu$ on the number of points at distance corresponding to level L_j – these are exactly the geometric weight constraints that make our near neighbor search primitives very efficient. Note that we are only considering level sets L_j for j at most $J = \lceil \log \frac{1}{\mu} \rceil$. We describe our implementation of importance sampling now.

Importance sampling: Suppose that one designs a sampling procedure that samples each point $\mathbf{p}_i$ with probability p_i and calculates the following estimator

$$Z = \sum_i \frac{\chi_i}{p_i} w_i$$

where $\chi_i = 1$ if $\mathbf{p}_i$ is sampled and $\chi_i = 0$ otherwise. Obviously, this estimator is an unbiased estimator for $n\mu^*$. So, if we can prove that this estimator has a relatively low variance, then by known techniques (repeating many times, averaging and taking the median) one can approximate μ^*, efficiently. It can be shown (see Claim 3.4.4) that if p_i's are proportional to w_i's (more specifically, we set $p_i \approx \frac{w_i}{n\mu}$) then the variance is low. This approach is known as **importance sampling**. In other words, we need to sample points with higher contribution, with higher probability.

If L_j's were known to the algorithm in the preprocessing phase, then for each j, one could have sampled points in L_j with probability $\approx \frac{1}{2^j n\mu}$. However, the query is not known in the preprocessing phase and hence geometric weight levels are not known beforehand.

Our approach is the following: for each j we sample the data set P with probability $\frac{1}{2^j n\mu}$. Then, we prepare a data structure (for this sampled data set) that can **recover** any sampled point with contribution $\approx 2^{-j}$ in the query procedure, efficiently and with high probability. Note that the number of points with contribution $\geq 2^{-j}$ is upper bounded by $2^j n\mu$. So, on average after the sub-sampling we expect to have at most $O(1)$ point from $L_1 \cup \ldots \cup L_j$. On the other hand, since Gaussian kernel is a decreasing function of distance, points in $L_{j+1} \cup \ldots \cup L_J$ are actually further than the query. Thus, our **recovery** problem can be seen as an instance of **near neighbor** problem. Therefore, we use the **locality sensitive hashing (LSH)** approach, which has been used in the literature for solving the approximate near neighbor problem.

Using Euclidean LSH for recovery: Now, we explain how one can use Euclidean LSH scheme to design a data structure to recover points from L_j in the corresponding sub-sampled data set.

We first present an informal and over-simplified version of LSH function used in [10]. Roughly speaking [10] presents the following result (see Lemma 3.4.1 for the formal statement):

Lemma 3.2.1 (Informal version of Lemma 3.4.1)**.** *For every r there exists a (locality sensitive) hash family such that, if $\mathbf{p}$ (a 'close' point) and $\mathbf{p}'$ (a 'far' point) are at distance r and $\geq c \cdot r$ (for some $c \geq 1$) of some point*

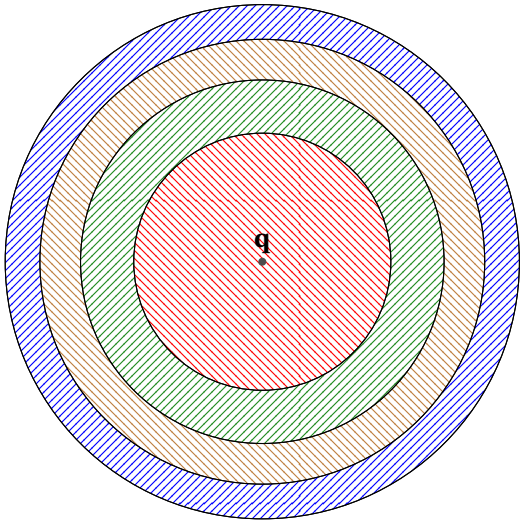


Figure 3.1: Illustration of distance levels induced by geometric weight levels. Areas marked with colors red, green, brown, blue and so on correspond to geometric weight levels $L_1, \ldots, L_4$ and so on.

q, *respectively, then if*

$$p := \mathbb{P}[h(\mathbf{p}) = h(\mathbf{q})],$$

then

$$\mathbb{P}[h(\mathbf{p}') = h(\mathbf{q})] \leq p^{(1-o(1))c^2}.$$

Now given a query $\mathbf{q}$, for every j we use Euclidean LSH to retrieve the points in L_j from a sample of the dataset where every point is included with probability $\frac{1}{2^j n\mu}$. We repeat the hashing process multiple times to ensure high probability of recovery overall, as in the original approach of [58]. However, the parameter setting and the analysis are different, since in the context of KDE we can exploit the geometric structure of the sampled dataset, namely upper bounds on the sizes of level sets L_j given in (3.3) above – we outline the parameter setting and analysis now.

On the other hand, geometric weight levels induce distance levels (see Definition 3.4.2 and Figure 3.1). Roughly speaking, for the Gaussian kernel if $\mathbf{p} \in L_j$ and $\mathbf{p}' \in L_i$, then

$$\frac{\|\mathbf{p}' - \mathbf{q}\|_2}{\|\mathbf{p} - \mathbf{q}\|_2} \approx \sqrt{\frac{i}{j}} =: c_{i,j}.$$

Recall that since we sampled the data set with probability $\frac{1}{2^j n\mu}$ then for every i we will have at most $\approx 2^{i-j}$ points from L_i in the sampled set, in expectation. In particular, most likely the sample does not contain points from level sets $i < j$. We instantiate Euclidean LSH from Lemma 3.2.1 with the 'near' distance r being the distance to the target level set L_j. Let p denote the probability that the query collides with a point in L_j. Now by Lemma 3.2.1 we upper bound the expected number of points from level sets $L_i, i > j$,

in the bucket of the query:

$$\sum_{i>j} 2^{i-j} \cdot p^{c_{i,j}^2}$$

We now select p (note that Lemma 3.2.1 allows flexibility in selecting p, which is achieved by concatenating hash functions; see Section 3.4 for the detailed analysis). We set p such that the number of points from each L_i in the bucket of the query is at most 1 for all $i > j$. For every such i, $2^{i-j} \cdot p^{\frac{i}{j}} \leq 1$ implies $p \leq \left(\frac{1}{2}\right)^{j-\frac{j^2}{i}}$, and hence we let

$$p = p_j = \min_{i>j} \left(\frac{1}{2}\right)^{j-\frac{j^2}{i}},$$

where we give the probability a subscript j to underscore that this is the setting for level set L_j.

On the other hand, note that since the point that we want to recover will be present in the query's bucket with probability p_j, we need to repeat this procedure $\widetilde{O}\left(\frac{1}{p_j}\right)$ times, to recover the point with high probability. This means that for every j the contribution of level set L_j to the query time will be $\widetilde{O}\left(\frac{1}{p_j}\right)$. Now, note that

$$\begin{aligned} \max_{j\in[J]} \log_2 \frac{1}{p_j} &= \max_{j\in[J]} \max_{i\in(j,J]} \left(j - \frac{j^2}{i}\right) \\ &= J \cdot \max_{j\in[J]} \frac{j}{J} \cdot \left(1 - \frac{j}{J}\right) \\ &= \frac{J}{4} \\ &= \frac{1}{4} \log \frac{1}{\mu}, \end{aligned} \tag{3.4}$$

implying a $(1/\mu)^{0.25}$ upper bound on the query time. This (informally) recovers the result mentioned in Theorem 3.2.1. Note that the space complexity of our data structure is no larger than the number of data points times the query time, i.e., $\approx n(1/\mu)^{0.25}$, since at every sampling rate we hash at most the entire dataset about $(1/\mu)^{0.25}$ times independently. The space complexity can also be bounded by $\widetilde{O}(1/\mu)$ by noting that the datasets for which we have the highest query time and hence many repetitions are in fact heavily subsampled versions of the input dataset. These bounds are incomparable, and the latter is preferable for large values of KDE value μ.

We used the Gaussian kernel in the informal description above to illustrate our main ideas, but the approach extends to a very general class of kernels. In particular, it gives improvements over all prior work on the KDE problem for shift invariant kernels (with the only exception that our results essentially match the results of [107], where an already very efficient algorithm with a polylogarithmic dependence on $1/\mu$ is presented). We present the detailed analysis of this approach in Section 3.4.

3.2.2 Data dependent algorithm (Section 3.5)

We note that the efficiency of our implementation of importance sampling relies heavily on the efficiency of near neighbor search primitive under density constraints. In this section we show how to use data-dependent techniques, i.e. data partitioning followed by the use of the more efficient Spherical LSH, to achieve significantly better results. Our approach builds on the exciting recent line of work on data-dependent near neighbor search [11, 12, 13], but the fact that we would like to optimally use the assumptions on the density of various spherical ranges that follow from assumptions on KDE value, the analysis turns out to be significantly more challenging. In particular, the core of our approach is a linear program that allows one to analyze the worst case evolution of densities during the hashing process. The analysis is presented in Section 3.5, Section 3.8 and Section 3.9. Since the analysis is somewhat involved, we present it for the case of the Gaussian kernel to simplify notation. We then provide a version of the key lemma for other kernels and state the corresponding results.

Theorem 3.2.2 (Informal version of Theorem 3.5.1)**.** *There exists an algorithm that, when K is the Gaussian kernel and $\mu^* := K(P,\mathbf{q})$, for $\epsilon \in (0,1)$ approximates μ^* to within a $(1\pm\epsilon)$ multiplicative factor, in expected time $\left(\frac{1}{\mu^*}\right)^{0.173+o(1)}$ and space $\min\{n\left(\frac{1}{\mu^*}\right)^{0.173+o(1)}, \left(\frac{1}{\mu^*}\right)^{1+o(1)}\}$.*

Our techniques extend to kernels beyond the Gaussian kernel (e.g., the exponential kernel, for which we obtain query time $\left(\frac{1}{\mu^*}\right)^{0.1+o(1)}$ and space $n\left(\frac{1}{\mu^*}\right)^{0.1+o(1)}$). We outline the extension in Section 3.5.

Recall that we need to preprocess a dataset P of n points $\mathbf{p}_1,\ldots,\mathbf{p}_n \in \mathbb{R}^d$ into a small space data structure that allows fast KDE queries, i.e., can quickly approximate, given $\mathbf{q} \in \mathbb{R}^d$, the quantity

$$\mu^* = K(P,\mathbf{q}) = \frac{1}{|P|}\sum_{\mathbf{p}\in P} \exp(-||\mathbf{p}-\mathbf{q}||_2^2/2). \tag{3.5}$$

Recall also that we assume knowledge of a quantity μ such that

$$\mu^* \le \mu \le 4\mu^*. \tag{3.6}$$

This is without loss of generality by a standard reduction – see Section 3.5, Remark 3.4.2. For simplicity of presentation, in this section we use a convenient rescaling of points so that

$$\mu^* = K(P,\mathbf{q}) = \frac{1}{|P|}\sum_{\mathbf{p}\in P} (1/\mu)^{-||\mathbf{p}-\mathbf{q}||_2^2/2}. \tag{3.7}$$

Note that this is simply a rescaling of the input points, namely multiplying every coordinate by $(\log(1/\mu))^{-1/2}$. This is for analysis purposes only, and the algorithm does not need to perform such a rescaling explicitly. We fix the query $\mathbf{q}$ for the rest of this section.

Densities of balls around query. Upper bounds on the number of points at various distances from the query point in dataset (i.e., densities of balls around the query) play a central part in our analysis. For any

$x \in (0, \sqrt{2})$ let

$$D_x(\mathbf{q}) := \{||\mathbf{p}-\mathbf{q}|| : \mathbf{p} \in P, ||\mathbf{p}-\mathbf{q}|| \gtrsim x\}, \tag{3.8}$$

denote the set of possible distances from $\mathbf{q}$ to points in the dataset P. Note that we are ignoring distances that are too close to x – this is for technical reasons that let us introduce some simplifications with respect to the analysis of [13] at the expense of a small constant loss in the exponent of the ultimate query time (see Section 3.5.3 for more discussion of this). When there is no ambiguity we drop $\mathbf{q}$ and x and we simply call it D. For any $y \in D$ we let

$$P_y(\mathbf{q}) := \{\mathbf{p} \in P : ||\mathbf{p}-\mathbf{q}|| \leq y\} \tag{3.9}$$

be the set of points at distance y from $\mathbf{q}$. Since for every $y > 0$

$$\begin{aligned}\mu^* = K(P,\mathbf{q}) &= \frac{1}{n}\sum_{\mathbf{p}\in P} \mu^{||\mathbf{p}-\mathbf{q}||_2^2/2} \\ &\geq \frac{\mu^{y^2/2}}{n}|P_y(\mathbf{q})|\end{aligned}$$

we get

$$|P_y(\mathbf{q})| \leq n\mu^* \cdot \left(\frac{1}{\mu}\right)^{\frac{y^2}{2}} \leq n \cdot \left(\frac{1}{\mu}\right)^{\frac{y^2}{2}-1}, \tag{3.10}$$

since $\mu^* \leq \mu$ by assumption.

We implement the same importance sampling strategy as in Section 3.2.1: sample the dataset at a geometric sequence of sampling rates, and for each such sampling rate use approximate near neighbor search primitives (in this case data dependent ones) to retrieve the relevant points (which are generally a few closest points to the query) from the sample. The rescaling of the input space (3.7) together with the assumption (3.6) implies that one essentially only needs to care about points $\mathbf{p} \in P$ such that

$$||\mathbf{p}-\mathbf{q}||_2 \approx x \text{ for some } x \in [0, \sqrt{2}).$$

This is because every $\mathbf{p} \in P$ such that $||\mathbf{p}-\mathbf{q}||_2 \geq \sqrt{2}$ contributes at most $(1/\mu)^{-||\mathbf{p}-\mathbf{q}||_2^2/2} \leq \mu \leq 4\mu^*$ by (3.6). This means that the contribution of such points can be approximated well by simply sampling every point with probability $\approx 1/n = 1/|P|$ and examining the entire sample – see Section C.3 for details. Therefore in the rest of this section (and similarly in its formal version, namely Section 3.5) we focus on the following single scale recovery problem:

> Given $x \in (0, \sqrt{2})$ and a sample $\widetilde{P}$ of the dataset P that includes every point with probability $\frac{1}{n} \cdot \left(\frac{1}{\mu}\right)^{1-\frac{x^2}{2}}$, recover all sampled points at distance at most $\approx x$ from the query.

Fix $x \in (0, \sqrt{2})$, and recall that $\widetilde{P}$ contains every point in P independently with probability $\frac{1}{n} \cdot \left(\frac{1}{\mu}\right)^{1-\frac{x^2}{2}}$. Note that by (3.10) for every $y \in (x, \sqrt{2}]$ the expected number of points at distance at most y from q that are

included in $\widetilde{P}$ is upper bounded by

$$n \cdot \left(\frac{1}{\mu}\right)^{\frac{y^2}{2}-1} \cdot \frac{1}{n} \cdot \left(\frac{1}{\mu}\right)^{1-\frac{x^2}{2}} \approx \left(\frac{1}{\mu}\right)^{\frac{y^2-x^2}{2}}. \tag{3.11}$$

What we defined so far is of course just a reformulation of our approach from Section 3.2.1, and indeed our data-dependent result follows the overall uniform sampling scheme. The difference comes in a much more powerful primitive for recovering data points at distance $\approx x$ from the query from the uniform sample. We describe this primitive now. In this development we start with the observation that underlies the work of [13] on data-dependent near neighbor search. Namely, one first observes that if the points in the sampled dataset $\widetilde{P}$ were uniformly random on the sphere (except of course for the actual points at distance $\approx x$ from the query $\mathbf{q}$), then instead of Euclidean LSH one could use random spherical caps to partition the dataset, leading to significantly improved performance. In order to leverage this observation, the work of [13] introduces the definition of a pseudo-random dataset (see Definition 3.3.3 below), gives an efficient procedure for decomposing any dataset into pseudorandom components and shows that the pseudorandom property is sufficiently strong to allow for about the same improvements as a random dataset does. Then their algorithm is a recursive process that partitions a given input dataset using random spherical caps, decomposes the resulting smaller datasets into pseudorandom components and recurses. Our algorithm follows this recipe, but the analysis turns out to be significantly more challenging due to the fact that we need to track the evolution of the densities of balls around the query during this recursive process. In what follows we state the necessary definitions and outline our algorithm.

The work of [13] introduces a key definition of a pseudorandom dataset (see Definition 3.3.3), which we reuse in our analysis and state here for convenience of the reader:

Definition 3.3.3 *(Restated) Let P be a set of points lying on $\mathcal{S}^{d-1}(o,r)$ for some $o \in \mathbb{R}^d$ and $r \in \mathbb{R}_+$. We call this sphere a* pseudo-random *sphere*[4]*, if $\nexists \mathbf{u}^* \in \mathcal{S}^{d-1}(o,r)$ such that*

$$\left|\left\{\mathbf{u} \in P : ||\mathbf{u}-\mathbf{u}^*|| \leq r(\sqrt{2}-\gamma)\right\}\right| \geq \tau \cdot |P|.$$

In other words, a dataset is pseudorandom on a sphere if at most a small fraction of this dataset can be captured by a spherical cap of nontrivially small volume. It turns out [13] that every dataset can be partitioned into pseudorandom components efficiently, so one can assume that the input dataset is pseudorandom. The significance of this lies in the fact that the power of Spherical LSH manifests itself on the points $\mathbf{p}$ at distance $\sqrt{2}-\gamma$ from the query essentially as well as on uniformly random points. Thus, if the fraction τ of 'violating' points is small, one now use Spherical LSH to partition the dataset into hash buckets and then recursive on the hash buckets, partition them into pseudorandom components and proceed recursively in this manner. Our algorithms follows this recipe, but the analysis introduces new techniques, as we describe below. We start by fixing some notation. Our algorithm (Algorithm 7) recursively constructs a tree $\mathcal{T}$ with alternating levels of SPHERICALLSH nodes and PSEUDORANDOMIFY nodes, which correspond to partitioning the dataset using locality sensitive hashing and extraction of dense components as per Definition 3.3.3 respectively. At every SPHERICALLSH node (Algorithm 8) we

[4]Whenever we say *pseudo-random sphere*, we implicitly associate it with parameter τ, γ which are fixed throughout the paper.

repeatedly generate subsets P' of the dataset P by sampling a Gaussian vector $g \sim N(0,1)^d$ and letting

$$P' \leftarrow \left\{\mathbf{p} \in P : \left\langle \frac{p-o}{R}, g \right\rangle \geq \eta \right\},$$

where R and o are the radius and center of the sphere that dataset P resides on, and $\eta = \omega(1)$ is an appropriately chosen parameter – we choose η to ensure that the collision probability of the query with a point at distance x from it is exactly $\mu^{1/T}$ for a parameter T (see line 16 of Algorithm 8). Crucially, we chose the parameter η to ensure that the size of the spherical cap is not too large. Specifically, for a parameter $T = \omega(1)$ that governs the depth of our recursive process we choose η to ensure that for every $\mathbf{p} \in P$ such that $||\mathbf{p} - \mathbf{q}||_2 \approx x$ one has

$$\mathbb{P}_{g \sim N(0,I)^d} \left[\left\langle \frac{\mathbf{p}-o}{R}, g \right\rangle \geq \eta | \left\langle \frac{\mathbf{q}-o}{R}, g \right\rangle \geq \eta \right] \approx \mu^{1/T},$$

where we assume for simplicity of presentation here that the query is on the sphere. The number of datasets $\widetilde{P}$ is chosen to be such that the query $\mathbf{q}$ collides with any given point $\mathbf{p}$ at distance $\approx x$ with high constant probability over all $O(T)$ levels of the tree $\mathcal{T}$. This means (see Section 3.6) that the expected number of datasets that the query $\mathbf{q}$ will be exploring is $(1/\mu)^{1/T}$. We limit the depth of the exploration process to $\approx 0.172 \cdot T$ (see line 27 of Algorithm 8), so that the QUERY algorithm (see Algorithm 10) explores at most $((1/\mu)^{1/T})^{0.172 \cdot T} = (1/\mu)^{0.172}$ leaf datasets in the tree $\mathcal{T}$. The main challenge lies in showing that these leaf datasets have small (nearly constant) expected size. In other words, we need to bound the effect of such a filtering process on the density of balls of various radius y around the query $\mathbf{q}$. Generally, the densities along any root to leaf path are decreasing because of two effects:

Truncation due to pseudorandom spheres: First effect that we consider is the condition that pseudo-randomness of spheres imply over the densities. Consider any query $\mathbf{q}$ and any pseudo-random sphere with radius r, and let ℓ be the distance from $\mathbf{q}$ to the center of the sphere. Let $\mathbf{q}'$ be the projection of the query on the sphere. Then, by pseudo-randomness of the sphere, we know that most of the points are orthogonal to $\mathbf{q}'$, i.e., have distance $\approx \sqrt{2}r$ from $\mathbf{q}'$ (see Lemma 3.3.1). However, we are interested in the condition that implies over the densities. Roughly speaking, the orthogonal points are at distance $c := \sqrt{\ell^2 + r^2}$. So we expect that the number of points at distance $\approx c$ will dominate the densities.

Claim 3.2.1 (Informal version of Claim 3.6.1)**.** *Suppose that a sphere with center o and radius r is pseudo-random. Then, if $\ell \approx ||\mathbf{q} - o||$, $c := \sqrt{\ell^2 + r^2}$ and for all y we let B_y be the number of points at distance y from $\mathbf{q}$ in the sphere. Then, the following conditions hold.*

$$\sum_{y \leq c - r\psi} B_y \leq \frac{\tau}{1-2\tau} \cdot \sum_{y \in (c-r\psi, c+r\psi)} B_y,$$

and

$$\sum_{y \geq c + r\psi} B_y \leq \frac{\tau}{1-2\tau} \cdot \sum_{y \in (c-r\psi, c+r\psi)} B_y,$$

where $\psi = o(1)$ is small factor.

Removing points due to Spherical LSH: The second phenomenon that reduces the densities is spherical LSH rounds. We set the size of the spherical cap as described above. Under this setting of size of spherical cap, the probability that a spherical cap conditioned on capturing the query, captures **p**, which is at distance y from **q**, is given by Claim 3.6.2, which is restated informally below.

Claim 3.2.2 (Informal version of Claim 3.6.2)**.** *Consider a sphere of radius r around point o, and let $\ell \approx ||\mathbf{q}-o||$. Also let* **p** *be a point on the sphere such that $y = ||\mathbf{p}-\mathbf{q}||$. Now, suppose that one generates a Gaussian vector g as in Algorithm 8. Then, we have*

$$\mathop{\mathbb{P}}_{g\sim N(0,1)^d}\left[\langle g, \frac{\mathbf{p}-o}{||\mathbf{p}-o||}\rangle \geq \eta | \langle g, \frac{\mathbf{q}-o}{||\mathbf{q}-o||}\rangle \geq \eta\right] \lesssim \exp_\mu\left(-\frac{4(r/x')^2-1}{4(r/y')^2-1}\cdot\frac{1}{T}\right).$$

where

- *η is such that $\frac{F(\eta)}{G(x'/r,\eta)} \approx \left(\frac{1}{\mu}\right)^{\frac{1}{T}}$ (see line 16 of Algorithm 8).*
- *$x' :=$* PROJECT*(x,ℓ,r) (see Definition 3.3.2).*
- *$y' :=$* PROJECT*(y,ℓ,r).*

We use Claim 3.2.1 and Claim 3.2.2 to bound the evolution of the density of various balls around the query **q** in the datasets constructed on the way from the root of the tree $\mathcal{T}$ down to a leaf.

Formally, we gather all necessary information about such a path in the definition of a *valid execution path* below:

Definition 3.6.3 (Valid execution path; slightly informal version) Let $R := (r_j)_{j=1}^J$ and $L := (\ell_j)_{j=1}^J$ for some positive values r_j's and ℓ_j's such that for all $j \in [J]$, $x \gtrsim |\ell_j - r_j|$. Also let D be as defined in (3.8). Then, for

$$A := (a_{y,j}), \quad y \in D, j \in [J] \cup \{0\} \qquad \text{(Intermediate densities)}$$
$$B := (b_{y,j}), \quad y \in D, j \in [J+1] \cup \{0\} \qquad \text{(Truncated intermediate densities)}$$

(L, R, A, B) is called a *valid execution path*, if the conditions below are satisfied for $\psi := o(1)$ and $c_j := \sqrt{r_j^2 + \ell_j^2}$ for convenience.

(1) Initial densities condition. The $a_{y,0}$ and $b_{y,0}$ variables are upper-bounded by the initial expected densities in the sampled dataset: for all $y \in D$

$$\sum_{y' \in [0,y] \cap D} a_{y',0} \le \min\left\{\exp_\mu\left(\frac{y^2 - x^2}{2}\right), \exp_\mu\left(\frac{1 - x^2}{2}\right)\right\}$$

and

$$\sum_{y' \in [0,y] \cap D} b_{y',0} \le \min\left\{\exp_\mu\left(\frac{y^2 - x^2}{2}\right), \exp_\mu\left(\frac{1 - x^2}{2}\right)\right\}$$

(2) Truncation conditions (effect of PSEUDORANDOMIFY). For any $j \in [J]$, for all $y \in D \setminus [\ell_j - r_j, \ell_j + r_j]$ one has $b_{y,j} = 0$ (density is zero outside of the range corresponding to the j-th sphere on the path; condition **(2a)**), for all $y \in D \cap [\ell_j - r_j, \ell_j + r_j]$ one has $b_{y,j} \le a_{y,j-1}$ (removing points arbitrarily **(2b)**) and

$$\sum_{y \in [0, c_j - \psi r_j] \cap D} b_{y,j} \le \frac{\tau}{1 - 2\tau} \cdot \sum_{y \in (c_j - \psi r_j, c_j + \psi r_j) \cap D} b_{y,j} \quad \text{(condition (2c))}$$

(3) LSH conditions. For every $j \in [J]$ and all $y \in [\ell_j - r_j, \ell_j + r_j] \cap D$

$$a_{y,j} \le b_{y,j} \cdot \exp_\mu\left(-\frac{4\left(\frac{r_j}{x'}\right)^2 - 1}{4\left(\frac{r_j}{y'}\right)^2 - 1} \cdot \frac{1}{T}\right)$$

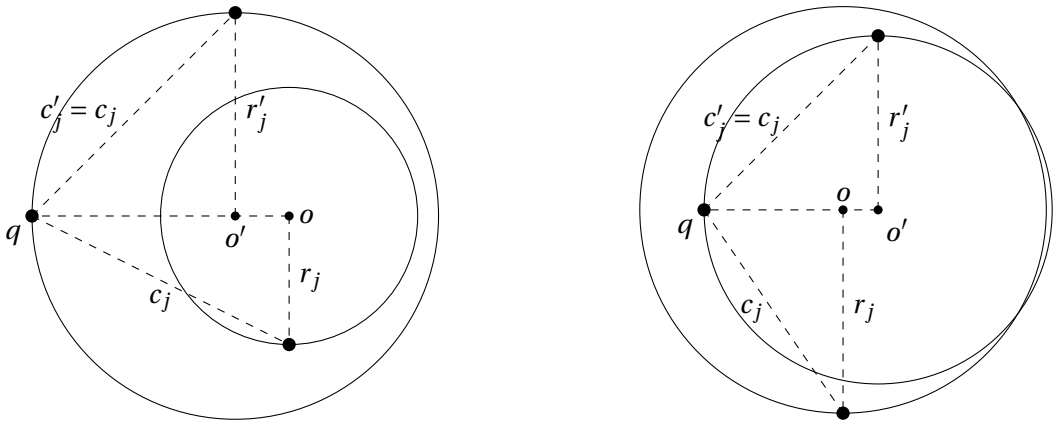
where $x' := \text{PROJECT}(x + \Delta, \ell_j, r_j)$ and $y' := \text{PROJECT}(y - \Delta/2, \ell_j, r_j)$. See Remark 3.6.1 below for a discussion about Δ factors.

(4) Terminal density condition. For any y such that $a_{y,J}$ is defined, $b_{y,J+1} \le a_{y,J}$.

Thus, our main goal is to show that

For every valid execution path (L, R, A, B) one has $\sum_y b_{y,J+1} = n^{o(1)}$.

The main challenge here is optimizing over sequences $(\ell_j, r_j)_{j=1}^J$ (distance to center of the sphere from $\mathbf{q}$ and the radius of the sphere). We perform this optimization in two steps, which we describe below.

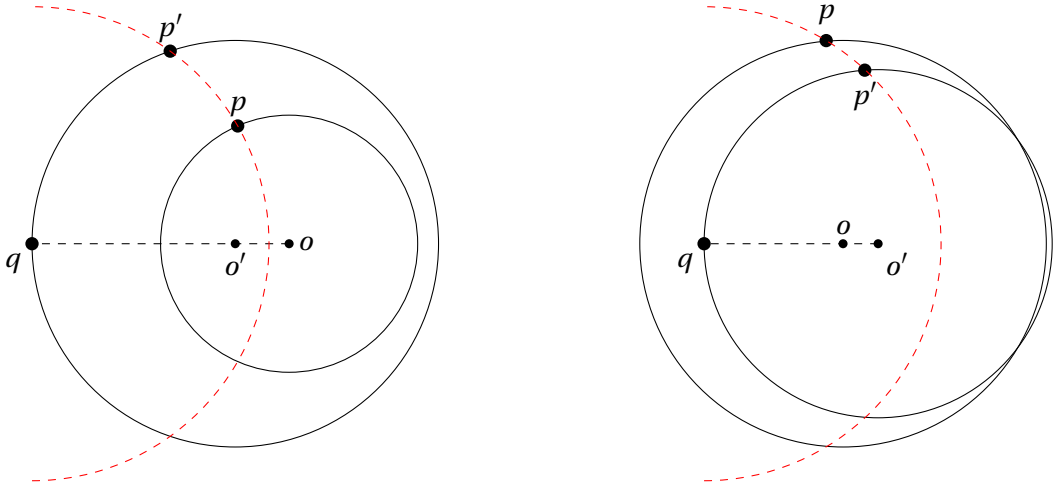


(a) When the query is outside of the sphere.

(b) When the query is inside the sphere

Figure 3.2: Converting a (non-zero-distance) sphere to its corresponding zero-distance sphere

Step 1. Suppose that there are two spheres such that the distance from the query to the orthogonal points for these spheres are the same. Also, assume that for the first sphere the query is not on the sphere, but for the second sphere the query is on the sphere (see Figure 3.2). Now, let **p** and **p**′ lie on the first and the second spheres, respectively. Moreover, assume that they have the same distance from the query (see Figure 3.3). Comparing the spherical LSH effect on these two spheres, we prove that **p** is removed with higher probability compared to **p**′ (see Claim 3.7.1). So, when the query is on the sphere, the densities are shrinking with a lower rate.



(a) When the query is outside of the sphere

(b) When the query is inside the sphere

Figure 3.3: Mapping points from a sphere to its corresponding zero-distance sphere.

Step 2. Now consider two spheres with different radii, and assume that query **q** lies on them at the same time. Due to less curvature on the larger sphere, after one round of spherical LSH on these two spheres, the densities are shrinking with a lower rate on this sphere compared to the smaller sphere (see Claim 3.7.2).

The following definition enables us to state our claims more efficiently.

Definition 3.6.5 *(Zero-distance and monotone path) Let* (L, R, A, B) *be an execution path defined in Definition 3.6.3. If for* $R = (r_j)_{j=1}^{J}$, r_j*'s are non-increasing in* j*, and* $L = R$*, then we say that* (L, R, A, B) *is a* zero-distance and monotone *execution path. When* $L = R$*, we usually drop* L*, and simply write* (R, A, B).

Now, using the two steps above, we can argue that for any valid execution path, we can find a zero-distance monotone execution path, with the same terminal densities and the same length (see Lemma 3.6.2 restated below for the convenience of the reader).

Lemma 3.6.2 *(Zero-distance and monotone path) For every valid execution path* (L, R, A, B) *(see Definition 3.6.3), there exists a* zero-distance and monotone *valid execution path* (R', A', B') *(see Definition 3.6.5) such that* $b'_{y,J+1} = b_{y,J+1}$ *for all* $y \in D$[5] *and* $|R'| = |R|$ *(i.e., the length of the paths are equal).*

The proof of the lemma (the formal version of the two steps mentioned above) is given in Section 3.7.

As mentioned before, we analyze the evolution of density of points in various distances. First, we define a grid of distances around query, which we use to properly round the distances of real spheres in the execution of algorithm. Second, instead of analyzing continuous densities, we define a new notion, called *discretized log-densities* (see Definition 3.6.7 below), for which we round densities to the discretized distances in a natural way, and for simplicity of calculations we take the log of these densities.

Definition 3.6.6 *(*x*-centered grid* Z_x*; restated) For every* $x \in (0, R_{max})$ *define the grid* $Z_x = \{z_I, z_{I-1}, \ldots, z_0\}$ *by letting* $z_I = x$*, letting* $z_{I-i} := (1+\delta_z)^i \cdot z_I$ *for all* $i \in [I]$ *and choosing the smallest integer* I *such that* $z_0 \geq R_{max}\sqrt{2}$.

Definition 3.6.7 *(Discretized log-densities* $f_{z_i,j}$*; restated) For any zero-distance monotone valid execution path* (R, A, B) *(as per Definition 3.6.3) with radii bounded by* R_{max} *and* $J = |R|$*, for all* $j \in [J]$ *let* k_j *be the index of the largest grid element which is not bigger than* $r_j \cdot (\sqrt{2} + \psi)$*, i.e.,*

$$r_j \cdot (\sqrt{2} + \psi) \in [z_{k_j}, z_{k_j - 1}) \tag{3.12}$$

and for every integer $i \in \{k_j, \ldots, I\}$ *define*

$$f_{z_i,j} := \log_{1/\mu}\left(\sum_{y \in D \cap [z_{i+1}, z_{i-1})} b_{y,j}\right) \tag{3.13}$$

Note that the variables $b_{y,j}$ *on the right hand side of* (3.13) *are the* $b_{y,j}$ *variables of the execution path* (R, A, B).

These two steps, allow us to analyze the evolution of densities over the course of time. In this section, we present an LP (see (3.33)) that its optimal cost bounds the query time of our algorithm. The main idea behind the linear program is to relax the notion of a zero-distance monotone path, which may involve only a small number of decreasing sphere radii, to a process that uses a grid $Z = Z_x$ of decreasing radii and possibly applies locality sensitive hashing at every such point (see the spherical LSH constraint in

[5] We need the final condition to argue that we have the same number of points remaining at the end.

(3.14) below), and applies pseudorandmification, i.e. ensures that the dataset is dominated by points at distance $z_j \approx \sqrt{2} r_j$ from the query (see the truncation constraints in (3.14) below). We note that the grid Z_x represents distances to points on the j-the sphere that are nearly orthogonal to the query, i.e. whose at distance $\approx \sqrt{2} r_j$, as opposed to the radii themselves. It is also important to note that the linear program is parameterized by two quantities: the target distance $x \in [0, \sqrt{2}]$ and a parameter j^* that indexes a point z_{j^*} in the grid Z_x. The quantity z_{j^*} should be thought of as the distance scale that contributes the most to query time, i.e. the band that the has the most number of points in the final densities (see non-empty range constraints in the linear program (3.14), as well as the similar calculation (3.4) in Section 3.2.1). To obtain our final bound on the query time, we enumerate over all x and $j^* \in Z_x$, upper bound the value of the corresponding $\mathrm{LP}(x, j^*)$ and take the maximum. Finally, we note that the intended LP solution is as follows. Consider a root to leaf path in the tree $\mathcal{T}$ constructed by $\textsc{PreProcess}(\widetilde{P}, x, \mu)$ that an invocation of $\textsc{Query}(\mathbf{q}, \mathcal{T}, x)$, and suppose that the sequence of radii of spheres traversed by $\textsc{Query}$ is exactly Z_x. Then letting α_j denote the number of LSH nodes that correspond to sphere with radius $r_j = z_j/\sqrt{2}$, divided by T, should intuitively give a feasible solution[6].

In Section 3.8 we will show in details why this LP formulation is enough to analyze the query time. Informally, this LP considers all possible root to leaf paths, and applies corresponding *truncation* and *spherical LSH* functions on the density and its cost is related to the length of root to leaf paths. We show in Section 3.8 that any execution path with large enough final densities gives a feasible solution to the linear program whose cost is (almost) equal to the length of the path divided by T. Thus, if we take any path with length more than $T \cdot \mathrm{OPT}(\mathrm{LP})$, the final densities are small.

Letting $Z := Z_x$ to simplify notation, we will consider I linear programs defined below in (3.33), enumerating over all $j^* \in [I]$, where we let $x' = x + \Delta$:

$$
\begin{aligned}
\mathrm{LP}(x, j^*): \quad & \max_{\alpha \geq 0} \sum_{j=1}^{j^*-1} \alpha_j && && (3.14)\\
& \forall y \in Z: g_{y,1} \leq \min\left\{ \frac{y^2 - x^2}{2}, 1 - \frac{x^2}{2} \right\} && \text{Density constraints} \\
& \text{for all } j < j^*, y \in Z, y < z_j: \\
& \quad g_{y,j} \leq g_{z_j,j} && \text{Truncation} \\
& \quad g_{y,j+1} \leq g_{y,j} - \frac{2\left(z_j/x\right)^2 - 1}{2\left(z_j/y\right)^2 - 1} \cdot \alpha_j && \text{Spherical LSH} \\
& \quad g_{z_{j^*}, j^*} \geq 0 && \text{Non-empty range constraint}
\end{aligned}
$$

The following claim is the main technical claim relating zero-distance monotone execution paths and the linear program (3.14):

[6] This statement is somewhat imprecise, and in fact is quite nontrivial to make fully formal – this is exactly what our algorithm achieves by introducing the notion of valid execution paths.

Claim 3.8.6 *(Feasible LP solution from an execution path; Restated) If integer J is such that $J > \frac{T}{1-10^{-4}}\mathrm{OPT(LP)}$ then, for all $y \le z_{j^*-1}$, $f_{y,J+1} < 7\delta_z$ for $j^* = k_J + 1$ (see Definition 3.6.7 for the definition of k_J).*

The proof of Claim 3.8.6 is somewhat delicate, and exploits specific properties of the (negative) log-density of the Gaussian kernel. In fact, one can construct rather simple kernels with non-decreasing log-density for which Claim 3.8.6 is false – we give an example in Figure 3.4a. Informally, we call a kernel well-behaved, if the log-densities after applying a few rounds of LSH (and corresponding truncations), are increasing up to some point and then they are decreasing. More formal description is given after the following paragraph.

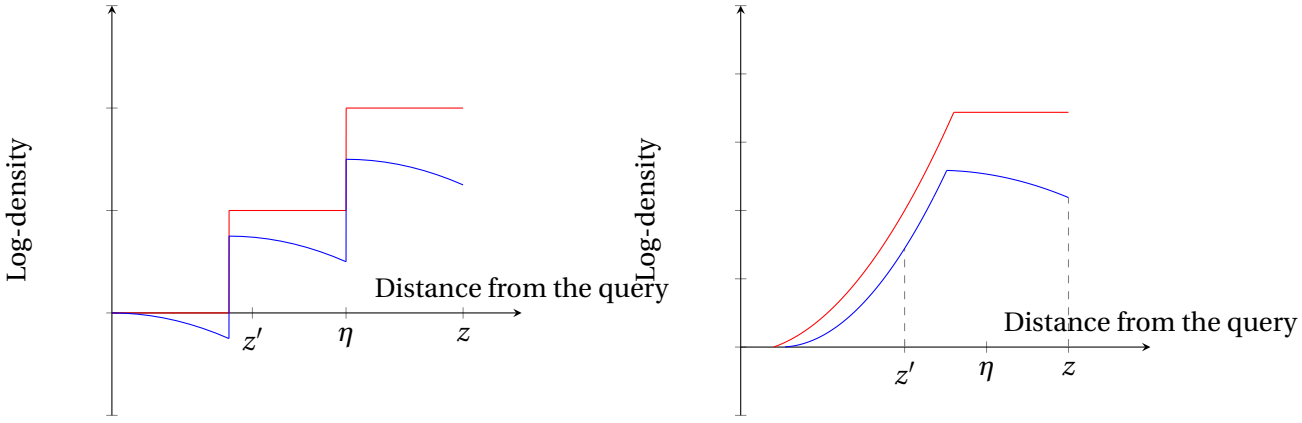
Intuitively, the reason is the difference between how the LP works and how the algorithm works. In the algorithm if we are running LSH on some sphere z we apply truncations based on distance z after each round of LSH (except the last step, for the intuition we can ignore this fact) and when we move to the next sphere z', the algorithm applies truncation to log-densities with respect to log-density at z'. However, the LP applies all the LSH rounds at once and then does truncation with respect to all bands from z to z'. Now, if some kernel is not well-behaved, say like the kernel depicted in Figure 3.4a then when the LP wants to move from z to z' it also truncates the log-densities with respect to the log-density at any $\eta \in (z', z)$. Then, for some η as shown in Figure 3.4a the log-density at some $\eta \in (z', z)$ is lower compared to the density at z and z'. Thus the log-densities in the LP shrink faster than the algorithm, which makes this approach not applicable to these set of kernels. However, for instance in the case of Gaussian kernel, the truncation with respect to log-densities at $\eta \in (z', z)$, do not impose a problem since the log-density at any $\eta \in (z', z)$ is larger than the minimum of densities at z and z'. This informally suggests that the evolution of the LP, can be seen as evolution of log-densities for well-behaved kernels, and thus can be used to analyze the run-time of the algorithm.

Now, we present a relatively more formally definition of well-behaved kernels. We say that a kernel $k(\mathbf{p}, \mathbf{q}) = \exp(-h(||\mathbf{p} - \mathbf{q}||_2))$ with the input space scaled so that $\exp(-h(\sqrt{2})) = \mu$ is well-behaved if for every integer $t \ge 1$, $x \in (0, \sqrt{2})$ and any sequence $c_1 \ge c_2 \ge \ldots \ge c_t \ge x$, such that

$$f(y) = \frac{y^2 - x^2}{2} - \sum_{s=1}^{t} \frac{2(c_s/x)^2 - 1}{2(c_s/y)^2 - 1} \cdot \frac{1}{T}$$

satisfies $f(\sqrt{2}c_t) > 0$, the following conditions hold. There exists $y^* \in (x, \sqrt{2}c_t]$ such that the function satisfies $f(y^*) = 0$ is monotone increasing on the interval $[y^*, \eta]$, where η is where the (unique) maximum of f on $(y^*, \sqrt{2}c_t]$ happens. See Fig. 3.4b for an illustration. Intuitively, a log-density h is well-behaved if the result of applying any amount of LSH on any collection of spheres to h results in a function with at most one maximum. This lets us control the structure of log-densities that arise after several iterations of LSH and truncation primitives in a valid execution path (and thus in a root to leaf path in $\mathcal{T}$ that a query $\mathbf{q}$ traverses).

We show in Section 3.8 (see Claim 3.8.4) that the Gaussian kernel is well behaved, and use this fact that prove Claim 3.8.6. We also show a similar claim for the class of kernels whose negative log density is concave (the exponential kernel is one example). This lets us extend our result to kernels beyond Gaussian (see Remark 3.5.1 in Section 3.5).



(a) A kernel, which is not well-behaved

(b) A well-behaved kernel

Figure 3.4: In both figures, the red curve and the blue curve represents the densities before and after running LSH rounds on sphere z, respectively. In the case of well-behaved kernels the density at any $\eta \in (z', z)$ is lower-bounded by the minimum of densities at z and z'. However, for a kernel which is not well-behaved, for instance for the η shown in the left figure, the density is lower than the density at z and z'.

On the other hand, we show numerically that the solution of the LP in (3.14) is upper bounded by 0.1718 for the Gaussian kernel. This is done in Section 3.9 by formulating the dual LP

$$\min \sum_{y \in Z} \left\{ \frac{y^2 - x^2}{2}, 1 - \frac{x^2}{2} \right\} r_{y,0} \tag{3.15}$$

such that :

$$\begin{aligned}
\forall j \in [j^* - 1], y \in Z, y < z_j : r_{y,j-1} - r_{y,j} + q_{y,j} = 0 \qquad & (g_{y,j}) \quad \text{Mass transportation} \\
\forall j \in [j^* - 1] : r_{z_j,j-1} - \sum_{x \in Z, x < z_j} q_{x,j} = 0 \qquad & (g_{z_j,j}) \quad \text{Max tracking} \\
\forall y \in Z, y < z_{j^*} : r_{y,j^*-1} = 0 \qquad & (g_{y,j^*}) \quad \text{Sink} \\
-\eta + r_{z_{j^*},j^*-1} = 0 \qquad & (g_{z_{j^*},j^*}) \quad \text{Terminal flow} \\
j \in [j^* - 1] : \sum_{y \in Z : y < z_j} \frac{2\left(z_j/x\right)^2 - 1}{2\left(z_j/y\right)^2 - 1} r_{y,j} \geq 1 \qquad & (\alpha_j) \\
r_{y,j}, q_{y,j} \geq 0 & \\
\eta \geq 0 &
\end{aligned}$$

and exhibiting a dual feasible solution of value ≈ 0.1716 for a fine grid of points Z_x and every x in a fine grid over $[0, \sqrt{2}]$. We also give an analytic upper bound of $\frac{x^2}{2}(1 - \frac{x^2}{2}) + 0.001$ on the value of the LP (3.14).

3.3 Preliminaries

We let $\mu^* \in (0,1]$ denote the kernel density of a dataset P in $\mathbb{R}^d$ at point $\mathbf{q} \in \mathbb{R}^d$:

$$\mu^* = K(P,\mathbf{q}) := \frac{1}{|P|}\sum_{\mathbf{p}\in P} K(\mathbf{p},\mathbf{q}).$$

3.3.1 Basic notation

Throughout the paper we assume that the points lie in a d-dimensional Euclidean space, $\mathbb{R}^d$. We let $\mathcal{S}^{d-1}$ denote the set of points on the unit radius sphere around the origin in $\mathbb{R}^d$. Also, for any $o \in \mathbb{R}^d$ and $R > 0$, we let $\mathcal{S}^{d-1}(o,R)$ to be the set of points on the sphere centered at o and radius R, and for any point $\mathbf{q} \in \mathbb{R}^d \setminus \{o\}$, the projection of $\mathbf{q}$ onto $\mathcal{S}^{d-1}(o,R)$ is defined as the closest point in $\mathcal{S}^{d-1}(o,R)$ to $\mathbf{q}$. For any pair of points $\mathbf{u},\mathbf{v} \in \mathbb{R}^d$, we let $||\mathbf{u}-\mathbf{v}||$ to be the Euclidean distance of $\mathbf{u}$ and $\mathbf{v}$.

For any integer J we define $[J] := \{1,2,\ldots,J\}$. For ease of notation in the rest of the paper, we let $\exp_\mu(a) := \left(\frac{1}{\mu}\right)^a$ and (abusing notation somewhat) let $\exp_2(a) = 2^a$ for any $a \in \mathbb{R}$.

3.3.2 $F(\eta)$ and $G(s,\eta,\sigma)$

In this section, we define notations and present results, which we later use to analyze the collision probability of spherical-LSH.

Lemma 3.3.1 (Lemma 3.1, [13])**.** *If for any $u \in \mathcal{S}^{d-1}$ we define*

$$F(\eta) := \mathop{\mathbb{P}}_{z\sim N(0,1)^d}\left[\langle z,u\rangle \geq \eta\right],$$

then, for $\eta \to \infty$

$$F(\eta) = e^{-(1+o(1))\cdot\frac{\eta^2}{2}}.$$

Lemma 3.3.2 (Lemma 3.2, [13])**.** *If for any $u,v \in \mathcal{S}^{d-1}$ such that $s := ||u-v||$, we define*

$$G(s,\eta,\sigma) := \mathop{\mathbb{P}}_{z\sim N(0,1)^d}\left[\langle z,u\rangle \geq \eta \text{ and } \langle z,v\rangle \geq \sigma\right],$$

then if $\sigma,\eta \to \infty$, and $\frac{\max\{\sigma,\eta\}}{\min\{\sigma,\eta\}} \geq \alpha(s)$, then one has

$$G(s,\eta,\sigma) = e^{-(1+o(1))\cdot\frac{\eta^2+\sigma^2-2\alpha(s)\eta\sigma}{2\beta^2(s)}},$$

where $\alpha(s) := 1-\frac{s^2}{2}$ and $\beta(s) := \sqrt{1-\alpha^2(s)}$.

Definition 3.3.1. *For ease of notation we also define*

$$G(s,\eta) := G(s,\eta,\eta).$$

3.3.3 Projection

Definition 3.3.2. *Let* $\mathbf{q}$ *be a point on* $\mathscr{S}^{d-1}(o,R_1)$ *and* $\mathbf{p}$ *be a point on* $\mathscr{S}^{d-1}(o,R_2)$*, such that* $y := ||q-p||$*. Now, if we define* $\mathbf{q}'$ *as the projection of* $\mathbf{q}$ *on* $\mathscr{S}^{d-1}(o,R_2)$*. Then, we define the following*

$$\textsc{Project}(y,R_1,R_2) := ||\mathbf{q}'-\mathbf{p}||.$$

Lemma 3.3.3. *For any* $R_1,R_2 \in \mathbb{R}_+$ *and* $o \in \mathbb{R}^d$ *assume that we have points* $\mathbf{q},\mathbf{p}$ *on spheres* $\mathscr{S}_1 := \mathscr{S}^{d-1}(o,R_1)$ *and* $\mathscr{S}_2 := \mathscr{S}^{d-1}(o,R_2)$*, respectively. Also, let* $x := ||\mathbf{p}-\mathbf{q}||$ *and let* $\mathbf{q}'$ *be the projection of point* $\mathbf{q}$ *on* $\mathscr{S}_2$*. Then we have the following*

$$\textsc{Project}(x,R_1,R_2) = ||\mathbf{q}'-\mathbf{p}|| = \sqrt{\frac{R_2}{R_1}\left(x^2-(R_2-R_1)^2\right)}.$$

The proof is deferred to Appendix C.1.

3.3.4 Pseudo-Random Spheres

Definition 3.3.3. *(Pseudo-random spheres) Let* P *be a set of points lying on* $\mathscr{S}^{d-1}(o,r)$ *for some* $o \in \mathbb{R}^d$ *and* $r \in \mathbb{R}_+$*. We call this sphere a* pseudo-random *sphere*[7]*, if* $\nexists \mathbf{u}^* \in \mathscr{S}^{d-1}(o,r)$ *such that*

$$\left|\left\{\mathbf{u} \in P : ||\mathbf{u}-\mathbf{u}^*|| \le r(\sqrt{2}-\gamma)\right\}\right| \ge \tau \cdot |P|.$$

As shown in [12, Section 6], it is possible to decompose a dataset P into pseudo-random components in time $\text{poly}\left(d,\gamma^{-1},\tau^{-1},\log|P|\right)\cdot|P|$. We present a slight modification of their argument using our notation in Appendix C.2. The following claim summarizes the properties of a pseudo-random sphere that we use later:

Claim 3.3.1. *If* P *is a set of points lying on* $\mathscr{S}^{d-1}(o,r)$ *for some* $o \in \mathbb{R}^d$ *and* $r \in \mathbb{R}_+$*, and* P *is a pseudo-random sphere (see Definition 3.3.3) then for any point* $\mathbf{q}'$ *on the sphere we have the following property*

$$\left|\left\{\mathbf{u} \in P : ||\mathbf{u}-\mathbf{q}'|| \le r(\sqrt{2}-\gamma)\right\}\right| \le \frac{\tau}{1-2\tau}\cdot\left|\left\{\mathbf{u} \in P : ||\mathbf{u}-\mathbf{q}'|| \in \left(r\left(\sqrt{2}-\gamma\right), r(\sqrt{2}+\gamma)\right)\right\}\right|,$$

and consequently,

$$\left|\left\{\mathbf{w} \in P : ||\mathbf{w}-\mathbf{q}'|| \in \left(r\left(\sqrt{2}-\gamma\right), r\left(\sqrt{2}+\gamma\right)\right)\right\}\right| = \Omega(|P|).$$

The proof is deferred to Appendix C.1.

[7] Whenever we say *pseudo-random sphere*, we implicitly associate it with parameter τ,γ which are fixed throughout the paper.

3.4 Kernel Density Estimation Using Andoni-Indyk LSH

In this section, we present an algorithm for estimating KDE, using the Andoni-Indyk LSH framework. In order to state the main result of this section for general kernels, we need to define a few notions first. Thus, we state the main result for Gaussian kernel in the following theorem, and then state the general result, Theorem 3.4.2, after presenting the necessary definitions.

Theorem 3.4.1. *Given a kernel $K(\mathbf{p},\mathbf{q}) := e^{-a||\mathbf{p}-\mathbf{q}||_2^2}$ for any $a>0$, $\epsilon = \Omega\left(\frac{1}{\text{polylog}n}\right)$, $\mu^* = n^{-\Theta(1)}$ and a data set of points P, using Algorithm 5 for preprocessing and Algorithm 6 for the query procedure, one can approximate $\mu^* := K(P,\mathbf{q})$ (see Definition 3.4.1) up to $(1\pm\epsilon)$ multiplicative factor, in time $\widetilde{O}\left(\epsilon^{-2}\left(\frac{1}{\mu^*}\right)^{0.25+o(1)}\right)$, for any query point $\mathbf{q}$. Additionally, the space consumption of the data structure is*

$$\min\left\{\epsilon^{-2}n\left(\frac{1}{\mu^*}\right)^{0.25+o(1)}, \epsilon^{-2}\left(\frac{1}{\mu^*}\right)^{1+o(1)}\right\}.$$

Throughout this section, we refer to Andoni-Indyk LSH's main result stated in the following lemma.

Lemma 3.4.1 ([10])**.** *Let $\mathbf{p}$ and $\mathbf{q}$ be any pair of points in $\mathbb{R}^d$. Then, for any fixed $r>0$, there exists a hash family $\mathcal{H}$ such that, if $p_{\text{near}} := p_1(r) := \mathbb{P}_{h\sim\mathcal{H}}[h(\mathbf{p}) = h(\mathbf{q}) \mid ||\mathbf{p}-\mathbf{q}|| \le r]$ and $p_{\text{far}} := p_2(r,c) := \mathbb{P}_{h\sim\mathcal{H}}[h(\mathbf{p}) = h(\mathbf{q}) \mid ||\mathbf{p}-\mathbf{q}|| \ge cr]$ for any $c\ge 1$, then*

$$\rho := \frac{\log 1/p_{\text{near}}}{\log 1/p_{\text{far}}} \le \frac{1}{c^2} + O\left(\frac{\log t}{t^{1/2}}\right),$$

for some t, where $p_{\text{near}} \ge e^{-O(\sqrt{t})}$ and each evaluation takes $d\,t^{O(t)}$ time.

Remark 3.4.1. *From now on, we use $t = \log^{2/3} n$, which results in $n^{o(1)}$ evaluation time and $\rho = \frac{1}{c^2} + o(1)$. In that case, note that if $c = O\left(\log^{1/7} n\right)$, then*

$$\frac{1}{\frac{1}{c^2} + O\left(\frac{\log t}{t^{1/2}}\right)} = c^2(1-o(1)).$$

Definition 3.4.1. *For a query $\mathbf{q}$, and dataset $P = \{\mathbf{p}_1,\ldots,\mathbf{p}_n\}$, we define*

$$\mu^* := K(P,\mathbf{q}) := \frac{1}{|P|}\sum_{\mathbf{p}\in P} K(\mathbf{p},\mathbf{q})$$

where for any $\mathbf{p}\in P$, $K(\mathbf{p},\mathbf{q})$ is a monotone decreasing function of $||\mathbf{q}-\mathbf{p}||$. Also, we define

$$w_i := K(\mathbf{p}_i,\mathbf{q}).$$

From now on, we assume that μ is a quantity such that

$$\mu^* \le \mu \tag{3.16}$$

We also use variable $J := \left\lceil \log_2 \frac{1}{\mu} \right\rceil$.

Definition 3.4.2 (Geometric weight levels)**.** *For any $j \in [J]$*

$$L_j := \left\{ \mathbf{p}_i \in P : w_i \in \left(2^{-j}, 2^{-j+1} \right] \right\}.$$

This implies corresponding distance levels (see Figure 3.1 and Figure 3.5), which we define as follows

$$\forall\, j \in [J] :\ r_j := \max_{s.t.\ f(r) \in \left(2^{-j}, 2^{-j+1}\right]} r.$$

where $f(r) := K(\mathbf{p}, \mathbf{p}')$ for $r = ||\mathbf{p} - \mathbf{p}'||$. Also define $L_{J+1} := P \setminus \cup_{j \in [J]} L_j$.[8]

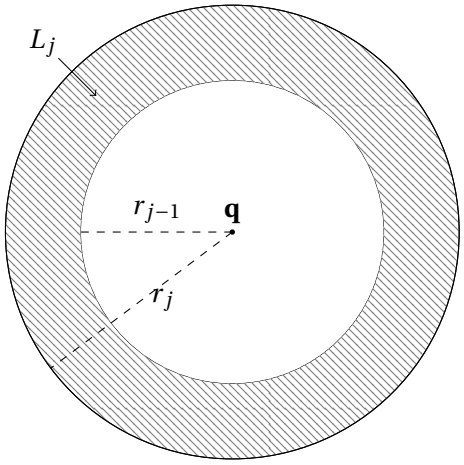


Figure 3.5: Illustration of definition of r_j's based on L_j's.

We start by stating basic bounds on collision probabilities under the Andoni-Indyk LSH functions in terms of the definition of geometric weight levels L_j (Definition 3.4.2):

Claim 3.4.1. *Assume that kernel K induces weight level sets, L_j's, and corresponding distance levels, r_j's (as per Definition 3.4.2). Also, for any query $\mathbf{q}$, any integers $i \in [J+1]$, $j \in [J]$ such that $i > j$, let $\mathbf{p} \in L_j$ and $\mathbf{p}' \in L_i$. And assume that $\mathscr{H}$ is an Andoni-Indyk LSH family designed for near distance r_j (see Lemma 3.4.1). Then, for any integer $k \geq 1$, we have the following conditions:*

1. $\mathbb{P}_{h^* \sim \mathscr{H}^k}\left[h^*(\mathbf{p}) = h^*(\mathbf{q}) \right] \geq p_{\text{near},j}^k$,
2. $\mathbb{P}_{h^* \sim \mathscr{H}^k}\left[h^*(\mathbf{p}') = h^*(\mathbf{q}) \right] \leq p_{\text{near},j}^{kc^2(1-o(1))}$,

where $c := c_{i,j} := \min\left\{ \frac{r_{i-1}}{r_j}, \log^{1/7} n \right\}$ (see Remark 3.4.1) and $p_{\text{near},j} := p_1(r_j)$ in Lemma 3.4.1.

Proof. If $\mathbf{p} \in L_j$ by Definition 3.4.2, we have

$$||\mathbf{q} - \mathbf{p}|| \leq r_j.$$

[8] One can see that $L_{J+1} = \{\mathbf{p}_i \in P : w_i \leq 2^{-J}\}$.

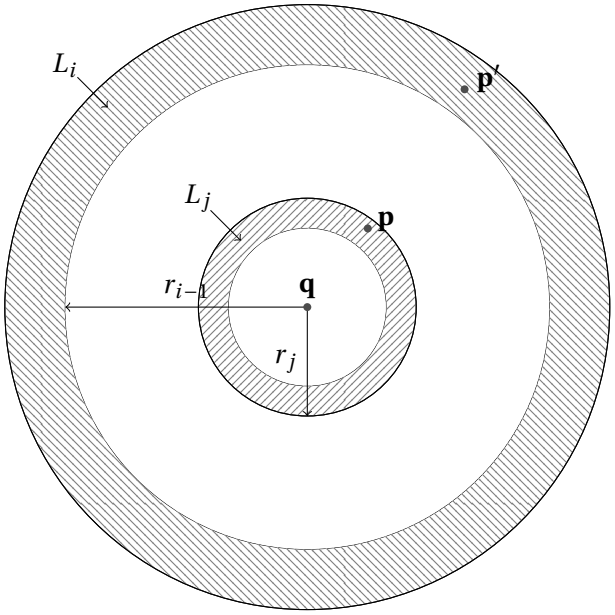


Figure 3.6: Illustration of r_j and r_{i-1} in terms of L_j and L_i.

Similarly using the fact that the kernel is decaying, for $\mathbf{p}' \in L_i$ we have

$$||\mathbf{q}-\mathbf{p}'|| \geq r_{i-1} \geq c \cdot r_j.$$

So, by Lemma 3.4.1 and Remark 3.4.1 the claim holds. Figure 3.6 shows an instance of this claim. □

Now, we prove an upper-bound on sizes of the geometric weight levels, i.e., L_j's (see Definition 3.4.2).

Lemma 3.4.2 (Upper bounds on sizes of geometric weight levels)**.** *For any $j \in [J]$, we have*

$$|L_j| \leq 2^j n\mu^* \leq 2^j n\mu.$$

Proof. For any $j \in [J]$ we have

$$\begin{aligned} n\mu \geq n\mu^* &= \sum_{\mathbf{p}\in P} K(\mathbf{p},\mathbf{q}) && \text{By Definition 3.4.1} \\ &\geq \sum_{i\in[J]} \sum_{\mathbf{p}\in L_i} K(\mathbf{p},\mathbf{q}) \\ &\geq \sum_{\mathbf{p}\in L_j} K(\mathbf{p},\mathbf{q}) \\ &\geq |L_j| \cdot 2^{-j} \end{aligned}$$

which proves the claim. □

Definition 3.4.3 (Cost of a kernel)**.** *Suppose that a kernel K induces geometric weight levels, L_j's, and corresponding distance levels, r_j's (see Definition 3.4.2). For any $j \in [J]$ we define* cost *of kernel K for weight*

level L_j *as*

$$\text{cost}(K, j) := \exp_2\left(\max_{i=j+1,\ldots,J+1}\left\lceil \frac{i-j}{c_{i,j}^2(1-o(1))}\right\rceil\right),$$

where $c_{i,j} := \min\left\{\frac{r_{i-1}}{r_j}, \log^{1/7} n\right\}$. *Also, we define the general* cost *of a kernel* K *as*

$$\text{cost}(K) := \max_{j\in[J]} \text{cost}(K, j).$$

Description of algorithm: The algorithm runs in J phases. For any $j \in [J]$, in the j'th phase, we want to estimate the contribution of points in L_j to $K(P, \mathbf{q})$. We show that it suffices to have an estimation of the number of points in L_j. One can see that if we sub-sample the data set with probability $\min\{\frac{1}{2^j n\mu}, 1\}$, then in expectation we get at most $O(1)$ points from L_i for any $i \le j$. Now, assume that a point $\mathbf{p} \in L_j$ gets sampled by sub-sampling, then we want to use Andoni-Indyk LSH to distinguish this point from other sub-sampled points, efficiently. Thus, we want to find the appropriate choice of k for the repetitions of Andoni-Indyk LSH (see Claim 3.4.1). Suppose that we call Claim 3.4.1 with some k (which we calculate later in (3.18)). Then we have

$$\mathop{\mathbb{P}}_{h^*\sim\mathscr{H}^k}\left[h^*(\mathbf{p}) = h^*(\mathbf{q})\right] \ge p_{\text{near},j}^k,$$

which implies that in order to recover point $\mathbf{p}$ with high probability, we need to repeat the procedure $\widetilde{O}\left(p_{\text{near},j}^{-k}\right)$ times. Another factor that affects the run-time of the algorithm is the number of points that we need to check in order to find $\mathbf{p}$. Basically, we need to calculate the number of points that hash to the same bucket as $\mathbf{q}$ under h^*'s. For this purpose, we use the second part of Claim 3.4.1, which bounds the collision probability of far points, i.e., points such as $\mathbf{p}' \in L_i$ for any $i > j$. Intuitively, for any point $\mathbf{p}' \in L_i$ for any $i > j$, by Claim 3.4.1 we have

$$\mathop{\mathbb{P}}_{h^*\sim\mathscr{H}^k}\left[h^*(\mathbf{p}') = h^*(\mathbf{q})\right] \le p^{kc^2(1-o(1))}$$

where $c := c_{i,j} := \min\left\{\frac{r_{i-1}}{r_j}, \log^{1/7} n\right\}$ and $p := p_{\text{near},j}$[9]. On the other hand, by Lemma 3.4.2, for $i = j+1, \ldots, J$ we have

$$|L_i| \le 2^i n\mu^* \le 2^i n\mu.$$

Then, one has the following bound,

$$\begin{aligned}
&\mathbb{E}\left[\left|\{\mathbf{p}' \in L_i : h^*(\mathbf{p}') = h^*(\mathbf{q})\}\right|\right] \\
&\le 2^i n\mu \cdot \frac{1}{2^j n\mu} \cdot p^{kc^2(1-o(1))} && \text{Sub-sampling and then applying LSH} \\
&= 2^{i-j} \cdot p^{kc^2(1-o(1))}. && (3.17)
\end{aligned}$$

[9]The indices are dropped for $c_{i,j}$ and $p_{\text{near},j}$ for ease of notation.

Since we have $O\left(\log \frac{1}{\mu}\right)$ geometric weight levels, then the expression in (3.17) for the worst i, bounds the run-time up to $O\left(\log \frac{1}{\mu}\right)$ multiplicative factor. In order to optimize the run-time up to $\widetilde{O}(1)$ multiplicative factors, we need to set k such that the expression in (3.17) gets upper-bounded by $O(1)$ for all $i > j$. So, in summary, for any fixed $j \in [J]$, we choose k such that any weight level L_i for $i \geq j$ contributes at most $\widetilde{O}(1)$ points in expectation to the hash bucket of the query, i.e., $h^*(\mathbf{q})$. One can see that we can choose k as follows

$$k := k_j := \frac{-1}{\log p} \cdot \max_{i=j+1,\ldots,J+1} \left\lceil \frac{i-j}{c_{i,j}^2(1-o(1))} \right\rceil. \tag{3.18}$$

For sampling the points in L_{J+1}, it suffices to sample points in the data set with probability $\frac{1}{n}$ (see line 15 in Algorithm 5), since the size of the sampled data set is small and there is no need to apply LSH. One can basically scan the sub-sampled data set.

Algorithm 5 Preprocessing

1: **procedure** PREPROCESS(P, ϵ)
2: ▷ P represents the set of data points
3: ▷ ϵ represents the precision of estimation
4: $K_1 \leftarrow \frac{C \log n}{\epsilon^2} \cdot \mu^{-o(1)}$ ▷ C is a universal constant
5: $J \leftarrow \left\lceil \log \frac{1}{\mu} \right\rceil$ ▷ We use geometric weight levels with base 2, see Definition 3.4.2
6: **for** $a = 1, 2, \ldots, K_1$ **do** ▷ $O(\log n/\epsilon^2)$ independent repetitions
7: **for** $j = 1, 2, \ldots, J$ **do** ▷ $J = \left\lceil \log \frac{1}{\mu} \right\rceil$ geometric weight levels
8: $K_2 \leftarrow 100 \log n \cdot p_{\text{near},j}^{-k_j}$
9: ▷ See Claim 3.4.1 and (3.18) for definition of $p_{\text{near},j}$ and k_j
10: $p_{\text{sampling}} \leftarrow \min\{\frac{1}{2^j n\mu}, 1\}$
11: $\widetilde{P} \leftarrow$ sample each element in P with probability p_{sampling}.
12: **for** $\ell = 1, 2, \ldots, K_2$ **do**
13: Draw a hash function from hash family $\mathscr{H}^{k_j}$ as per Claim 3.4.1 and call it $H_{a,j,\ell}$
14: Run $H_{a,j,\ell}$ on $\widetilde{P}$ and store non-empty buckets
15: $\widetilde{P}_a \leftarrow$ sample each element in P with probability $\frac{1}{n}$
16: Store $\widetilde{P}_a$ ▷ Set $\widetilde{P}_a$ will be used to recover points beyond L_{J+1}

Algorithm 6 Query procedure

procedure QUERY($P, \mathbf{q}, \epsilon, \mu$)
  ▷ P represents the set of data points
  ▷ ϵ represents the precision of estimation
  $K_1 \leftarrow \frac{C \log n}{\epsilon^2} \cdot \mu^{-o(1)}$ ▷ C is a universal constant
  $J \leftarrow \left\lceil \log \frac{1}{\mu} \right\rceil$ ▷ We use geometric weight levels with base 2, see Definition 3.4.2
  for $a = 1, 2, \ldots, K_1$ **do** ▷ $O(\log n/\epsilon^2)$ independent repetitions
    for $j = 1, 2, \ldots, J$ **do** ▷ $J = \left\lceil \log \frac{1}{\mu} \right\rceil$ geometric weight levels
      $K_2 \leftarrow 100 \log n \cdot p_{\text{near},j}^{-k_j}$ ▷ See Claim 3.4.1 and (3.18) for definition of $p_{\text{near},j}$ and k_j
      for $\ell = 1, 2, \ldots, K_2$ **do**
        Scan $H_{a,j,\ell}(q)$ and recover points in L_j
    Recover points from L_{J+1} in the sub-sampled dataset, $\widetilde{P}_a$.
    $S \leftarrow$ set of all recovered points in this iteration
    for $\mathbf{p}_i \in S$ **do**
      $w_i \leftarrow K(\mathbf{p}_i, \mathbf{q})$
      if $\mathbf{p}_i \in L_j$ for some $j \in [J]$ **then**
        $p_i \leftarrow \min\{\frac{1}{2^J n \mu}, 1\}$,
      else if $\mathbf{p}_i \in P \setminus \cup_{j \in [J]} L_j$ **then**
        $p_i \leftarrow \frac{1}{n}$
    $Z_a \leftarrow \sum_{\mathbf{p}_i \in S} \frac{w_i}{p_i}$

Now, we present the main result of this section.

Theorem 3.4.2 (Query time)**.** *For any kernel K, the expected query-time of the algorithm is equal to $\widetilde{O}\left(\epsilon^{-2} n^{o(1)} \cdot \text{cost}(K)\right)$.*

Assuming Theorem 3.4.2, we prove Theorem 3.4.1.

Proof of Theorem 3.4.1: We first start by proving the query time bound and then we prove the space consumption of the data structure, and the guarantee over the precision of the estimator is given in Claim 3.4.4.

Proof of the query time bound: We calculate the cost of Gaussian kernel $e^{-a\|\mathbf{x}-\mathbf{y}\|_2^2}$. First, we present the weight levels and distance levels induced by this kernel. As per Definition 3.4.1, let

$$\mu^* := K(P, \mathbf{q}) = \sum_{\mathbf{p} \in P} e^{-a\|\mathbf{p}-\mathbf{q}\|_2^2}.$$

By Definition 3.4.2, one has

$$\begin{aligned} L_j &:= \left\{\mathbf{p}_i \in P : w_i \in \left(2^{-j}, 2^{-j+1}\right]\right\} \\ &= \left\{\mathbf{p}_i \in P : ||\mathbf{p}_i - \mathbf{q}||_2 \in \left[\sqrt{\frac{(j-1)\ln 2}{a}}, \sqrt{\frac{j\ln 2}{a}}\right)\right\}, \end{aligned}$$

which immediately translates to $r_j := \sqrt{\frac{j\ln 2}{a}}$ for all $j \in [J]$. Also, we for all $i \in [J+1]$, $j \in [J]$ such that $i > j$, we have

$$\begin{aligned} c_{i,j} &:= \min\left\{\frac{r_{i-1}}{r_j}, \log^{1/7} n\right\} \\ &= \min\left\{\sqrt{\frac{i-1}{j}}, \log^{1/7} n\right\} \end{aligned}$$

At this point, one can check that

$$\max_{j\in[J]} \max_{i=j+1,\ldots,J+1} \left\lceil \frac{i-j}{c_{i,j}^2(1-o(1))} \right\rceil = (1+o(1))\frac{1}{4}\log\frac{1}{\mu},$$

Therefore, the cost of Gaussian kernel is

$$\mathrm{cost}(K) = \left(\frac{1}{\mu}\right)^{(1+o(1))\frac{1}{4}}.$$

Now, invoking Theorem 3.4.2, the statement of the claim about the query time holds.

Proof of the space bound: First, since the query time is bounded by $\epsilon^{-2}\left(\frac{1}{\mu^*}\right)^{0.25+o(1)}$, then the number of hash functions used is also bounded by the same quantity. This implies that the expected size of the space needed to store the data structure prepared by the preprocessing algorithm is $\epsilon^{-2} n \left(\frac{1}{\mu^*}\right)^{0.25+o(1)}$, since for each hash function we are hashing at most n points (number of points in the dataset).

For the other bound, we need to consider the effect of sub-sampling the data set. Fix $j \in [J]$. In the phase when we are preparing the data structure to recover points from L_j, we sub-sample the data set with probability $\min\{\frac{1}{2^j n\mu}, 1\}$, and then we apply $\widetilde{O}\left(p_{\mathrm{near},j}^{-k_j}\right)$ hash functions to this sub-sampled data set. Since

$$k_j = \frac{-1}{\log p} \cdot \max_{i=j+1,\ldots,J+1} \left\lceil \frac{i-j}{c_{i,j}^2(1-o(1))} \right\rceil,$$

by (3.18), where $p = p_{\mathrm{near},j}$, we have

$$p_{\mathrm{near},j}^{-k_j} = \exp_2\left(\max_{i=j+1,\ldots,J+1} \left\lceil \frac{i-j}{c_{i,j}^2(1-o(1))} \right\rceil - j\right). \tag{3.19}$$

At the same time, the expected size of the sampled dataset is bounded by $n \cdot \min\{\frac{1}{2^j n\mu}, 1\} \le \frac{1}{\mu} \cdot 2^{-j}$. Putting this together with the equation above, we get that the expected size of the dataset constructed for level L_j is upper bounded by

$$\frac{1}{\mu} \exp_2 \left(\max_{i=j+1,\dots,J+1} \left\lceil \frac{i-j}{c_{i,j}^2 (1-o(1))} \right\rceil - j \right). \tag{3.20}$$

Now for every $i = j+1, \dots, J$ such that $c_{i,j} = \sqrt{\frac{i-1}{j}}$ one has

$$\max_{i=j+1,\dots,J+1} \left\lceil \frac{i-j}{c_{i,j}^2 (1-o(1))} \right\rceil - j = \max_{i=j+1,\dots,J+1} \left\lceil j \cdot \frac{i-j}{(i-1)(1-o(1))} \right\rceil - j \le o(J),$$

and for the other values of i we have $\max_{i=j+1,\dots,J+1} \left\lceil \frac{i-j}{\log^{1/7} n (1-o(1))} \right\rceil - j \le o(J)$ as well. Putting this together with (3.20) and multiplying by $J = O(\log(1/\mu)) = \mu^{-o(1)}$ to account for the number of choices $j \in [J]$, we get the second bound for the expected size of the data structure $\epsilon^{-2} \left(\frac{1}{\mu^*}\right)^{1+o(1)}$.

Proof of the precision of the estimator: First, we prove the following claim, which guarantees high success probability for recovery procedure.

Claim 3.4.2 (Lower bound on probability of recovering a sampled point)**.** *Suppose that we invoke Algorithm 5 with (P, ϵ). Suppose that in line 11 of Algorithm 5, when $k = k^*$ and $j = j^*$, we sample some point $\mathbf{p} \in L_{j^*}$. We claim that with probability at least $1 - \frac{1}{n^{10}}$, there exists $\ell^* \in [K_2]$ such that $H_{k^*,j^*,\ell^*}(\mathbf{p}) = H_{k^*,j^*,\ell^*}(\mathbf{q})$.*

Proof. By Claim 3.4.1 we have

$$\mathbb{P}_{h^* \sim \mathscr{H}^k} \left[h^*(\mathbf{p}) = h^*(\mathbf{q}) \right] \ge p_{\text{near},j}^{k_j}.$$

Now note that we repeat this process for $K_2 = 100 \log n \cdot p_{\text{near},j}^{-k_j}$ times. So any point $\mathbf{p}$ which is sampled from band L_{j^*} is recovered in at least one of the repetitions of phase $j = j^*$, with high probability. □

Now, we argue that the estimators are unbiased (up to small inverse polynomial factors)

Claim 3.4.3 (Unbiasedness of the estimator)**.** *For every $\mu^* \in (0,1)$, every $\mu \ge \mu^*$, every $\epsilon \in (\mu^{10}, 1)$, every $\mathbf{q} \in \mathbb{R}^d$, estimator Z_a for any $a \in [K_1]$ constructed in* QUERY$(P, \mathbf{q}, \epsilon, \mu)$ *(Algorithm 6) satisfies the following:*

$$(1 - n^{-9}) n\mu^* \le \mathbb{E}[Z_a] \le n\mu^*$$

Proof. Let $\mathscr{E}$ be the event that every sampled point is recovered and let $Z := Z_a$ (see line 19 in Algorithm 6). By Claim 3.4.2 and union bound, we have

$$\mathbb{P}[\mathscr{E}] \ge 1 - n^{-9}$$

We have that $\mathbb{E}[Z] = \sum_{i=1}^{n} \frac{\mathbb{E}[\chi_i]}{p_i} w_i$ with $(1-n^{-9})p_i \le \mathbb{E}[\chi_i] \le p_i$, where we now define $\chi_i = 1$ if point $\mathbf{p}_i$ is sampled and recovered in the phase corresponding to its weight level, and $\chi_i = 0$ otherwise. Thus

$$(1-n^{-9})n\mu^* \le \mathbb{E}[Z] \le n\mu^*. \tag{3.21}$$

□

Remark 3.4.2. *We proved that our estimator is unbiased*[10] *for* **any choice** *of $\mu \ge \mu^*$. Therefore if $\mu \ge 4\mu^*$, by Markov's inequality the estimator outputs a value larger than μ at most with probability $1/4$. We perform $O(\log n)$ independent estimates, and conclude that μ is higher than μ^* if the median of the estimated values is below μ. This estimate is correct with high probability, which suffices to ensure that we find a value of μ that satisfies $\mu/4 < \mu^* \le \mu$ with high probability by starting with some $\mu = n^{-\Theta(1)}$ (since our analysis assumes $\mu^* = n^{-\Theta(1)}$) and repeatedly halving our estimate (the number of times that we need to halve the estimate is $O(\log n)$ assuming that μ is lower bounded by a polynomial in n, an assumption that we make).*

Claim 3.4.4 (Variance bounds)**.** *For every $\mu^* \in (0,1)$, every $\epsilon \in (\mu^{10}, 1)$, every $\mathbf{q} \in \mathbb{R}^d$, using estimators Z_a, for $a \in [K_1]$ constructed in* QUERY*$(P, \mathbf{q}, \epsilon, \mu)$ (Algorithm 6), where $\mu/4 \le \mu^* \le \mu$, one can output a $(1 \pm \epsilon)$-factor approximation to μ^*.*

Proof. By Claim 3.4.3 and noting that $Z \le n^2\mu^*$, where the worst case (equality) happens when all the points are sampled and all of them are recovered in the phase of their weight levels. Therefore,

$$\mathbb{E}[Z|\mathscr{E}] \cdot \mathbb{P}[\mathscr{E}] + n^2\mu^*(1 - \mathbb{P}[\mathscr{E}]) \ge \mathbb{E}[Z].$$

Also, since Z is a non-negative random variable, we have

$$\mathbb{E}[Z|\mathscr{E}] \le \frac{\mathbb{E}[Z]}{\mathbb{P}[\mathscr{E}]} \le \frac{n\mu^*}{\mathbb{P}[\mathscr{E}]} = n\mu^*(1 + o(1/n^9))$$

Then, we have

$$\begin{aligned}
\mathbb{E}[Z^2] &= \mathbb{E}\left[\left(\sum_{\mathbf{p}_i \in P} \chi_i \frac{w_i}{p_i}\right)^2\right] \\
&= \sum_{i \ne j} \mathbb{E}\left[\chi_i \chi_j \frac{w_i w_j}{p_i p_j}\right] + \sum_{i \in [n]} \mathbb{E}\left[\chi_i \frac{w_i^2}{p_i^2}\right] \\
&\le \sum_{i \ne j} w_i w_j + \sum_{i \in [n]} \frac{w_i^2}{p_i} \mathbb{I}[p_i = 1] + \sum_{i \in [n]} \frac{w_i^2}{p_i} \mathbb{I}[p_i \ne 1] \\
&\le \left(\sum_i w_i\right)^2 + \sum_{i \in [n]} w_i^2 + \max_i \left\{\frac{w_i}{p_i} \mathbb{I}[p_i \ne 1]\right\} \sum_{i \in [n]} w_i \\
&\le 2n^2(\mu^*)^2 + \max_{j \in [J], \mathbf{p}_i \in L_j} \{w_i 2^{j+1}\} n\mu \cdot n\mu^* \\
&\le 4n^2\mu^2 \qquad \text{Since } \mu^* \le \mu
\end{aligned}$$

[10] Up to some small inverse polynomial error.

and

$$\mathbb{E}[Z^2|\mathscr{E}] \leq \frac{\mathbb{E}[Z^2]}{\mathbb{P}[\mathscr{E}]} \leq n^2\mu^{2-o(1)}(1+o(1/n^9))$$

Now, since $\mu \leq 4\mu^*$, in order to get a $(1\pm\epsilon)$-factor approximation to μ^*, with high probability, it suffices to repeat the whole process $K_1 = \frac{C\log n}{\epsilon^2}\cdot\mu^{-o(1)}$ times, where C is a universal constant.

Suppose we repeat this process m times and $\bar{Z}$ be the empirical mean, then:

$$\begin{aligned}
\mathbb{P}[|\bar{Z}-\mu^*| \geq \epsilon n\mu^*] &\leq \mathbb{P}[|\bar{Z}-\mathbb{E}[Z]| \geq \epsilon\mu^* - |\mathbb{E}[Z]-n\mu^*|] \\
&\leq \mathbb{P}[|\bar{Z}-\mathbb{E}[Z]| \geq (\epsilon-n^{-9})n\mu^*] \\
&\leq \frac{\mathbb{E}[\bar{Z}^2]}{(\epsilon-n^{-9})^2(n^2\mu^*)^2} \\
&\leq \frac{1}{m}\frac{16n^2(\mu^*)^2}{(\epsilon-n^{-9})^2(n^2\mu^*)^2}
\end{aligned}$$

Thus by picking $m = O(\frac{1}{\epsilon^2})$ and taking the median of $O(\log(1/\delta))$ such means we get a $(1\pm\epsilon)$-approximation with probability at least $1-\delta$ per query. □

All in all, we proved the expected query time bound, the expected space consumption and the precision guarantee in the statement of the theorem. □

Now, we calculate the cost of kernel for t-student kernel.

t-student kernel ($\frac{1}{1+||\mathbf{x}-\mathbf{y}||_2^t}$): We directly calculate distance levels induced by this kernel as follows

$$r_j = \sqrt[t]{2^j-1}$$

which implies that for all $i \in [J+1]$, $j \in [J]$ such that $i > j$,

$$\begin{aligned}
c_{i,j} &:= \min\left\{\frac{r_{i-1}}{r_j}, \log^{1/7} n\right\} \\
&= \min\left\{\sqrt[t]{\frac{2^{i-1}-1}{2^j-1}}, \log^{1/7} n\right\}.
\end{aligned}$$

Now, one can check that

$$\max_{j\in[J]}\max_{i=j+1,\ldots,J+1}\left\lceil\frac{i-j}{c_{i,j}^2(1-o(1))}\right\rceil = \frac{\log\frac{1}{\mu}}{\log^{2/7} n}(1+o(1)).$$

Thus, we have

$$\text{cost}(K) = \mu^{-o(1)}.$$

We note that this matches the result of [107] up to the difference between $\mu^{-o(1)}$ and $\log(1/\mu)$ terms. The $\mu^{-o(1)}$ dependence comes from the fact that we used the LSH of [10], and the dependence can be improved to $\log(1/\mu)$ by using the hash family of [111], for instance.

Exponential Kernel ($e^{-\|x-y\|_2}$) The distance levels induced by the kernel are given by $r_j = j\log(2)$ for $j \in [J]$. Hence, we get that $c_{ij} = \min\left\{\frac{r_{i-1}}{r_j}, \log^{1/7} n\right\} = \min\{\frac{i-1}{j}, \log^{1/7} n\}$. If $i > j\log^{1/7} n + 1$ then the cost is increasing in $i > 0$ becomes:

$$\text{cost}(K, j) = \exp_2\left(\frac{J+1-j}{\log^{2/7} n}\right) \leq \exp_2\left(\frac{J}{\log^{2/7} n}\right) = \mu^{-o(1)}.$$

Thus, for the rest we will assume that $i \leq j\log^{1/7} n + 1$, and we need to find the maximum over j of

$$(1+o(1)) \max_{i=j+1,\ldots,J+1} \left\lceil \frac{j^2((i-1)-(j-1))}{(i-1)^2} \right\rceil$$

Setting $x = i-1$ and $A = j-1$, we optimize the function $\frac{(x-A)}{x^2}$ for $x \geq A+1$. We get that the optimal value is attained for $i^*(j) = \max\{\min\{2j-1, J+1\}, j+1\}$. We distinguish three cases:

1. $j = 1$: then $i^* = 2$ and we get $\text{cost}(K, 1) = \mu^{-o(1)}$
2. $j > \frac{J+2}{2}$: then the maximum over i is $\frac{j^2(J+1-j)}{j^2}(1+o(1))$, and the optimal choice of j is $j^* = \frac{2(J+1)}{3}$. We thus get
$$\max_{j>\frac{J+2}{2}}\{\text{cost}(K, j)\} = \mu^{-(1+o(1))\frac{4}{27}}$$
3. $j \leq \frac{J+2}{2}$: then the maximum over i is $\frac{j^2}{4(j-1)}(1+o(1))$ and the optimal choice for j is $j^* = \frac{J+2}{2}$. We thus get
$$\max_{j\leq\frac{J+2}{2}}\{\text{cost}(K, j)\} = \mu^{-(1+o(1))\frac{1}{8}}.$$

Overall, the worst-case cost is attained for $i^* = J$ and $j^* = \frac{2J}{3}$ and yields

$$\text{cost}(K) = \mu^{-(1+o(1))\frac{4}{27}}.$$

Proof of Theorem 3.4.2: One should note that the query time of our approach depends on the number of times that we hash the query and the number of points that we check, i.e., the number of points that collide with the query. First, we analyze the number of points colliding with the query. We Fix $j \in [J]$, so, we want to estimate the contribution of points in L_i to $K(P, \mathbf{q})$. We consider 3 cases:

Case 1. $i \leq j$: Note that we have $|L_i| \leq 2^i n\mu$ and note that in j'th phase, we sample the data set with rate $\min\{\frac{1}{2^j n\mu}, 1\}$. Thus, we have at most $1 = O(1)$ sampled points from L_i in expectation.

Case 2. $i = j+1,\ldots,J$**:** Again, note that by Lemma 3.4.2, $|L_i| \leq 2^i n\mu$, and the sampling rate is $\min\{\frac{1}{2^j n\mu}, 1\}$. Thus, we have at most 2^{i-j} sampled points from L_i in expectation. Now, we need to analyze the effect of LSH. Note that we choose LSH function such that the near distance is r_j (see Claim 3.4.1). Also, note that as per (3.18), we use

$$k := k_j := \frac{-1}{\log p_{\text{near},j}} \cdot \max_{i=j+1,\ldots,J+1} \left\lceil \frac{i-j}{c_{i,j}^2(1-o(1))} \right\rceil.$$

as the number of concatenations. Now, we have the following collision probability for $\mathbf{p} \in L_i$ using Claim 3.4.1

$$\mathop{\mathbb{P}}_{h^* \in \mathscr{H}^k} \left[h^*(\mathbf{p}) = h^*(\mathbf{q}) \right] \leq p^{kc^2(1-o(1))},$$

where $c := c_{i,j} := \min\left\{ \frac{r_{i-1}}{r_j}, \log^{1/7} n \right\}$ and $p := p_{\text{near},j}$ for ease of notation. This implies that the expected number of points from weight level L_i in the query hash bucket is at most

$$2^{i-j} \cdot p^{kc^2(1-o(1))} = \widetilde{O}(1)$$

by the choice of k.

Case 3. points in L_{J+1}**:** We know that we have n points, so after sub-sampling, we have at most $\frac{1}{2^j\mu}$ points from this range, remaining in expectation. For any $\mathbf{p} \in L_{J+1}$, note that $||\mathbf{p}-\mathbf{q}|| \geq c \cdot r_j$ for $c := c_{J+1,j} := \min\left\{ \frac{r_J}{r_j}, \log^{1/7} n \right\}$. Then,

$$\mathop{\mathbb{P}}_{h^* \in \mathscr{H}^k} \left[h^*(\mathbf{p}) = h^*(\mathbf{q}) \right] \leq p^{kc^2(1-o(1))},$$

which implies that the expected number of points form this range in the query hash bucket is at most

$$\frac{1}{2^j\mu} \cdot p^{kc^2(1-o(1))} = 2^{J-j} \cdot p^{kc^2(1-o(1))} = \widetilde{O}(1)$$

by the choice of k_j.

All in all, we prove that each weight level L_i for $i \in [J+1]$ contribute at most $\widetilde{O}(1)$ points to the hash bucket of query. Now, we need to prove a bound on the number of times we evaluate our hash function. One should note that by the choice of k_j in (3.18) we have

$$k_j = \widetilde{O}(1)$$

which basically means that we only concatenate $\widetilde{O}(1)$ LSH functions. Thus, we the evaluation time of $h^*(q)$ for any $h^* \in \mathscr{H}^k$ is $\widetilde{O}(n^{o(1)})$, by Remark 3.4.1. On the other hand, note that for recovering the points in L_{J+1} we just sub-sampled the data set with probability $\frac{1}{n}$ so in expectation we only scan 1 point. So in total, since we repeat this for all $j \in [J]$ and $J = \lceil \log \frac{1}{\mu} \rceil$, by the choice of K_1 and K_2 assigned in lines 4 and 8 of Algorithm 5, respectively, the claim holds. □

3.5 Improved algorithm via data dependent LSH

In this section, we improve the algorithm presented in the previous section using data dependent LSH approach for the Gaussian kernel. Consider a data set $P \subset \mathbb{R}^d$, a positive real number a, and a query $\mathbf{q} \in \mathbb{R}^d$. Let

$$\mu^* := K(P, \mathbf{q}) = \sum_{\mathbf{p} \in P} e^{-a||\mathbf{p}-\mathbf{q}||_2^2}$$

denote the KDE value at the query $\mathbf{q} \in \mathbb{R}^d$ of interest, and for the rest of the paper suppose that the algorithm is given a parameter μ that satisfies the following property

$$\mu^* \leq \mu. \tag{3.22}$$

We prove the following main result in the rest of the paper.

Theorem 3.5.1. *Given a kernel $K(\mathbf{p}, \mathbf{q}) := e^{-a||\mathbf{p}-\mathbf{q}||_2^2}$ for any $a > 0$, $\epsilon = \Omega\left(\frac{1}{\text{polylog}n}\right)$, $\mu^* = n^{-\Theta(1)}$ and a data set of points P, there exists a preprocessing algorithm and a corresponding query algorithm that one can approximate $\mu^* := K(P, \mathbf{q})$ (see Definition 3.4.1) up to $(1 \pm \epsilon)$ multiplicative factor, in time $\widetilde{O}\left(\epsilon^{-2}\left(\frac{1}{\mu^*}\right)^{0.173+o(1)}\right)$, for any query point $\mathbf{q}$. Additionally, the space consumption of the data structure is*

$$\min\left\{\epsilon^{-2} n \left(\frac{1}{\mu^*}\right)^{0.173+o(1)}, \epsilon^{-2}\left(\frac{1}{\mu^*}\right)^{1+c+o(1)}\right\}.$$

for a small constant $c = 10^{-3}$.

Proof. First, in Section 3.5.1 we present the main primitives in the preprocessing phase (Algorithms 7, 8 and 9) and prove the space bound in Lemma 3.5.2. The standard outer algorithm is presented in Appendix C.3 for completeness. The main query primitive in query algorithm is presented in Section 3.5.3, and the query time is proved in Section 3.6 in Lemma 3.6.1. The correctness proof (precision of the estimator) is rather standard and similar to the correctness proof in Section 3.4 and is given in Appendix C.3 for completeness. □

Remark 3.5.1. *Although we present the analysis for the Gaussian kernel, our techniques can be used for other kernels such as the exponential kernel as well. We do not present the full analysis to simplify presentation of our main result for the Gaussian kernel, but provide proofs of key lemmas in Appendix C.6. Specifically, we present the equivalent of Claims 3.8.4 and 3.8.5, which underly our LP analysis, for kernels whose negative log density is concave (this includes the exponential kernel $\exp(-||x||_2)$). Our dual solution presented in Section 3.9 gives an upper bound of ≈ 0.1 on the value of the corresponding LP. Replacing the parameter α^* in the algorithms presented in this section with 0.1 thus yield an data structure for KDE with the exponential kernel with query time $\widetilde{O}\left(\epsilon^{-2}\left(\frac{1}{\mu^*}\right)^{0.1+o(1)}\right)$ and space consumption $\epsilon^{-2} n \left(\frac{1}{\mu^*}\right)^{0.1+o(1)}$ for the exponential kernel.*

In order to simplify notation we apply the following normalization without loss of generality: For any point in $\mathbf{p} \in P \cup \{\mathbf{q}\}$, let $\mathbf{p}' := \sigma\mathbf{p}$ and $\sigma := \sqrt{\frac{2a}{\log(1/\mu)}}$ such that a point $\mathbf{p}'$ at distance $\sqrt{2}$ from the query $\mathbf{q}'$

contributes exactly μ to the kernel. In other words, we assume by convenient scaling that

$$K(P,\mathbf{q}) = \frac{1}{n}\sum_{\mathbf{p}'}(\mu)^{||\mathbf{p}'-\mathbf{q}'||_2^2/2}.$$

and to lighten notation we will assume that $\sigma = 1$, i.e. points are already properly scaled. For $x \in (0,\sqrt{2})$, let $\widetilde{P}$ be the dataset obtained from P by including every point independently with probability $\min\left\{\frac{1}{n}\cdot\left(\frac{1}{\mu}\right)^{1-\frac{x^2}{2}},1\right\}$. We state these conditions in a compact way as follows and use them in the rest of the paper.

Assumption 3.5.1. *We have the followings*

- $P \subset \mathbb{R}^d$ *and* $|P| = n$.
- $\mathbf{q} \in \mathbb{R}^d$.
- $\mu^* := K(P,\mathbf{q})$
- $\frac{1}{\mu^*} = n^{\Omega(1)}$
- μ *is such that* $\mu^* \leq \mu$.
- $\mu = n^{-\Theta(1)}$.
- *The points are scaled so that* $K(\mathbf{p},\mathbf{q}) = \mu^{\frac{||\mathbf{p}-\mathbf{q}||^2}{2}}$.
- $\widetilde{P}$ *is obtained by independently sub-sampling elements of* P *with probability* $\min\left\{\frac{1}{n}\cdot\left(\frac{1}{\mu}\right)^{1-\frac{x^2}{2}},1\right\}$, *for some* $x \in (0,\sqrt{2})$, *which is clear from the context.*

In this section we design a data structure that allows preprocessing $\widetilde{P}$ as above using small space such that every point at distance at most x from any query $\mathbf{q}$ is recovered with probability at least 0.8 (see Lemma C.3.1).

In what follows we present our preprocessing algorithm (Algorithm 7) in Section 3.5.1, the query algorithm (Algorithm 10) in Section 3.5.3 as well as proof of basic bounds on their performance in the same sections. Our main technical contribution is the proof of the query time bound. This proof relies on a novel linear programming formulation that lets us bound the evolution of the density of points around the query q as the query percolates does the tree $\mathcal{T}$ of hash buckets produced by PREPROCESS. This analysis is given in Section 3.6, with the main supporting technical claims presented in Section 3.8.

3.5.1 Preprocessing algorithm and its analysis

Our preprocessing algorithm is recursive. At the outer level, given the sampled dataset $\widetilde{P}$ as input, the algorithm hashes $\widetilde{P}$ into buckets using Andoni-Indyk Locality sensitive hashing. The goal of this is to ensure that with high probability all hash buckets that a given query explores are of bounded diameter,

while at the same time ensuring that any close point $\mathbf{p}$ hashes together with $\mathbf{q}$ in at least one of the hash buckets with high constant probability. The corresponding analysis is presented in Sections 3.5.3 and 3.6.

Our main tool in partitioning the data set into (mostly) low diameter subsets is an Andoni-Indyk Locality Sensitive Hash family. Such a family is provided by Lemma 3.4.1, which was our main tool in obtaining the non-adaptive KDE primitives in Section 3.4, and Corollary 3.5.1 below. We restate the lemma below for convenience of the reader:

Lemma 3.4.1 *([10]) (Restated) Let* $\mathbf{p}$ *and* $\mathbf{q}$ *be any pair of points in* $\mathbb{R}^d$*. Then, for any fixed* $r > 0$*, there exists a hash family* $\mathscr{H}$ *such that, if* $p_{\text{near}} := p_1(r) := \mathbb{P}_{h\sim\mathscr{H}}[h(\mathbf{p}) = h(\mathbf{q}) \mid ||\mathbf{p}-\mathbf{q}|| \leq r]$ *and* $p_{\text{far}} := p_2(r,c) := \mathbb{P}_{h\sim\mathscr{H}}[h(\mathbf{p}) = h(\mathbf{q}) \mid ||\mathbf{p}-\mathbf{q}|| \geq cr]$ *for any* $c > 1$*, then*

$$\rho := \frac{\log 1/p_{\text{near}}}{\log 1/p_{\text{far}}} \leq \frac{1}{c^2} + O\left(\frac{\log t}{t^{1/2}}\right),$$

for some t*, where* $p_{\text{near}} \geq e^{-O(\sqrt{t})}$ *and each evaluation takes* $d\,t^{O(t)}$ *time.* One should also recall Remark 3.4.1, which ensures $n^{o(1)}$ evaluation time, with appropriate choice of t in Lemma 3.4.1.

Corollary 3.5.1. *Let* α *be a constant, and let* $x \in (0,\sqrt{2})$ *and* y *be such that* $y \geq x$*. Then, there exists a hash family* $\mathscr{H}$ *such that for any points* $\mathbf{q} \in \mathbb{R}^d$*,* $\mathbf{p}$ *and* $\mathbf{p}'$*, where* $||\mathbf{p}-\mathbf{q}|| \leq x$ *and* $||\mathbf{p}'-\mathbf{q}|| \geq y$*, we have the following conditions*

- $\mathbb{P}_{h\sim\mathscr{H}}\left[h(\mathbf{q}) = h(\mathbf{p})\right] \geq \mu^{\alpha}$
- $\mathbb{P}_{h\sim\mathscr{H}}\left[h(\mathbf{q}) = h(\mathbf{p}')\right] \leq \mu^{\alpha c^2(1-o(1))}$

where $c := \min\{\frac{y}{x}, \log^{1/7} n\}$*, and we call such a hash family a* (α, x, μ)*-AI hash family.*

Our preprocessing algorithm is given below. It simply hashes the dataset several times independently using an Andoni-Indyk LSH family and calls SPHERICAL-LSH (Algorithm 8 below) on the buckets. The hashing is repeated several times to ensure that the query collides with any given close point with high probability in at least one of the hashings. Overall PREPROCESS simply reduces the diameter of the dataset, whereas most of the work is done by SPHERICAL-LSH, defined below.

Algorithm 7 PREPROCESS: $\widetilde{P}$ is the subsampled data-set, x is the target distance to recover

procedure PREPROCESS($\widetilde{P}, x, \mu$)
  Add a root w_0 to the recursion tree $\mathscr{T}$
  $w_0.P \leftarrow \widetilde{P},\ w_0.level \leftarrow 0,\ w_0.g \leftarrow 0$ ▷ $g = 0$ since this node uses Euclidean LSH
  if $x > \sqrt{2}$ **then return** $\widetilde{P}$ ▷ In that case the expected size of $\widetilde{P}$ is small
  $\alpha \leftarrow 10^{-4}$ ▷ Choice of α affects hash bucket diameter, see Lemma 3.5.1
  $K_1 = 100\left(\frac{1}{\mu}\right)^{\alpha}$ ▷ The number of repetitions of the first round of hashing
  for $j = 1, 2, \ldots, \lceil K_1 \rceil$ **do**
    Pick h_j from a (α, x, μ)-AI hash family, $\mathscr{H}$ ▷ See Definition 3.5.1
    $B \leftarrow$ set of non-empty hash buckets by hashing points in P using h_j.
    for each $b \in B$ **do**
      Add a node v as a child of w_0 in recursion tree $\mathscr{T}$
      $v.P \leftarrow b,\ v.level \leftarrow 0,\ v.g \leftarrow 0$
      $v.o \leftarrow$ any point in bucket b
      SPHERICAL-LSH(v, x, μ)
  return $\mathscr{T}$

We will use the following basic upper bound on the Euclidean diameter of LSH buckets:

Lemma 3.5.1 (Diameter bound for Andoni-Indyk LSH buckets)**.** *Under Assumption 3.5.1, suppose that $\mathscr{H}$ is a (α, x, μ)-AI hash family (see Corollary 3.5.1), for some constant α and let $c := \min\left\{\frac{R_{\text{diam}}}{x}, \log^{1/7} n\right\}$ for some $R_{\text{diam}} \geq \sqrt{2}$, then if $\alpha c^2 = 2 + \Omega(1)$ then one has*

(a) $\mathbb{E}_{h\sim\mathscr{H},\widetilde{P}}\left[\left|\left\{\mathbf{p} \in \widetilde{P} : ||\mathbf{p}-\mathbf{q}|| \geq R_{\text{diam}} \text{ and } h(\mathbf{p}) = h(\mathbf{q})\right\}\right|\right] \leq \left(\frac{1}{\mu}\right)^{2-\alpha c^2(1-o(1))}$

(b) *and consequently*

$$\mathop{\mathbb{P}}_{h\sim\mathscr{H},\widetilde{P}}[\textit{diameter of } h^{-1}(\mathbf{q}) \cap \widetilde{P} \textit{ is larger than } R_{\text{diam}}] \leq \left(\frac{1}{\mu}\right)^{2-\alpha c^2(1-o(1))}.$$

Proof. One has for every $R_{\text{diam}} \geq \sqrt{2}$

$$\mathbb{E}_{\widetilde{P}}\left[\left|\left\{\mathbf{p} \in \widetilde{P} : ||\mathbf{p}-\mathbf{q}|| \geq R_{\text{diam}}\right\}\right|\right] \leq \mathbb{E}_{\widetilde{P}}\left[\left|\widetilde{P}\right|\right] = n \cdot \frac{1}{n}\left(\frac{1}{\mu}\right)^{1-\frac{x^2}{2}} = \left(\frac{1}{\mu}\right)^{1-\frac{x^2}{2}}$$

Taking the expectation with respect to the hash function h, we get,

$$\begin{aligned}\mathbb{E}_{h\sim\mathscr{H},\widetilde{P}}\left[\left|\left\{\mathbf{p} \in \widetilde{P} : ||\mathbf{p}-\mathbf{q}|| \geq R_{\text{diam}} \text{ and } h(\mathbf{p}) = h(\mathbf{q})\right\}\right|\right] &\leq \left(\frac{1}{\mu}\right)^{1-\frac{x^2}{2}-\alpha c^2(1-o(1))} \\ &\leq \left(\frac{1}{\mu}\right)^{2-\alpha c^2(1-o(1))},\end{aligned}$$

establishing **(a)**. Claim **(b)** now follows by applying Markov's inequality since $\alpha c^2 = 2 + \Omega(1)$. □

Now, we establish a constant upper bound on the diameter of data set after the Andoni-Indyk LSH round. Since we have $\mu = n^{-\Theta(1)}$, and $\alpha = 10^{-4}$ as per line 5 of Algorithm 7, by Lemma 3.5.1 if we let R_{diam} be a large enough constant, then one has

$$\mathop{\mathbb{P}}_{h}[\text{diameter of } h^{-1}(\mathbf{q}) \cap \widetilde{P} \text{ is larger than } R_{diam}] \leq n^{-20} \tag{3.23}$$

Let $\mathscr{E}_{diam}$ denote the event that all Andoni-Indyk hash buckets that the query hashes to have diameter bounded by R_{diam}. We have, combining the failure event over sampling of $\widetilde{P}$ (over-sampling by a factor more than $O(\log n)$) with (3.23) that $\mathbb{P}[\bar{\mathscr{E}}] \leq 2n^{-20} \leq n^{-19}$. Conditioned on $\mathscr{E}$ buckets that the query hashes have diameter bounded by R_{diam}. Now, if we take any point in the data set and consider a ball of radius $R_{\text{max}} := 2R_{\text{diam}}$, using the triangle inequality, it contains all the points of this hash bucket. This ensures that all the spheres in the recursion tree have radius bounded by $R_{\text{max}} = \Theta(1)$.

Corollary 3.5.2 (Bounded diameter spheres)**.** *All the spheres that the query scan in the algorithm have radius bounded by* $R_{max} = \Theta(1)$.[11]

We are now ready to present our main preprocessing primitive SPHERICAL-LSH, given as Algorithm 8 below. The input to the algorithm is a node in the recursion tree $\mathscr{T}$ created by recursive invocations of SPHERICAL-LSH. Every such node v is annotated with a dataset $v.P$, a radius $v.r$ of a ball enclosing the dataset, the center $v.o$ of that ball and a level, $v.level$, initially set to 0 for the root of the tree $\mathscr{T}$ that is created by PREPROCESS. SPHERICAL-LSH then proceeds as follows. First it calls the PSEUDORANDOMIFY procedure (Algorithm 9 below). This procedure partitions the input dataset $v.P$ into subsets that are pseudorandom as per Definition 3.3.3. A similar procedure was used in the work of [13] on space/query time tradeoffs for nearest neighbor search. Intuitively, a dataset is pseudorandom if the points belong to a thin spherical shell and furthermore do not concentrate on any spherical cap in this shell (appropriately defined). These pseudorandom datasets are added to the recursion tree $\mathscr{T}$ as children of v. SPHERICAL-LSH then generates random subsets of these pseudorandom spheres defined by random spherical caps, adds these datasets to the recursion tree $\mathscr{T}$ and recursively calls itself until a depth budget T (see line 3 below) is exhausted. Note that the radius of spherical caps generated depends on the distance x' from the projected query point to the target near point (which is assumed to be at distance x from the query). Note that since the query is not available at the preprocessing stage, the algorithm prepares data structures for all possible values of x' (see line 14 in Algorithm 8 below). Note that the value of x' is passed down the recursion tree. In the section below, we set the parameters that we use in the algorithms.

3.5.2 Parameter settings

- $\gamma = \frac{1}{\log\log\log n}$ and $\tau = \frac{1}{10}$ are the parameters used for pseudo-random spheres (see Definition 3.3.3) in Algorithm 9.
- $\alpha^* = 0.172$ (see Section 3.9), $T = \sqrt{\log n}$ and $J = \min\left\{\alpha^* \cdot T, \left(\frac{x^2}{2}\left(1 - \frac{x^2}{2}\right) + 10^{-4}\right) \cdot T\right\}$ are parameters to bound the depth of the recursion tree.

[11] Since we did not use any density constraints other than the upper bound of n on the number of points, this corollary applies for all spheres in the Algorithm.

- $\delta = \exp(-(\log\log n)^C)$ for some large enough constant C, is a parameter used for partitioning point in a ball to discrete spheres of radii multiplies of δ (see Algorithm 9)
- $\delta' = \exp(-(\log\log n)^C)$ for some large enough constant C, is a parameter for rounding x''s to x'''s (see lines 18 and 19 in Algorithm 10)
- $R_{\min} = 10^{-5}$ is a lower bound on the radius of spheres that we process further, i.e., we stop whenever the radius becomes less than $R_{\min}$.
- $\Delta = 10^{-20}$ is a tiny constant. For a discussion about Δ see Remark 3.6.1.
- $\alpha = 10^{-4}$ is a parameter used for the Andoni-Indyk LSH round (see Algorithm 7)
- $\delta_z = 10^{-6}$ is a parameter used in discretizing continuous densities in Definition 3.6.7.
- $\delta_x = 10^{-8}$ is used for defining a grid over $(0, \sqrt{2})$, such that for any x from this grid we prepare the data structure to recover points from $[x - \delta, x)$ (see Algorithm 13).

Algorithm 8 SPHERICAL-LSH: x is the target distance to recover, v is the node in recursion tree (corresponds to a subset of the dataset)

1: **procedure** SPHERICAL-LSH(v, x, μ)
2: $\gamma \leftarrow \frac{1}{\log\log\log n}$
3: $T \leftarrow \sqrt{\log n}$
4: $U \leftarrow$ PSEUDORANDOMIFY(v, γ)
5: **for** $w \in U$ **do**
6: Add w as a child of v in recursion tree $\mathcal{T}$
7: $P \leftarrow w.P$ ▷ The dataset of w
8: $R \leftarrow w.r$ ▷ Radius of the sphere of w
9: **if** $R < R_{min}$ **continue**
10: $\delta' \leftarrow \exp(-(\log\log n)^C)$
11: $o \leftarrow w.o$ ▷ Center of sphere of w
12: $W \leftarrow \left\{ \lfloor \frac{\Delta-\delta}{\delta'} \rfloor \cdot \delta', \left(\lfloor \frac{\Delta-\delta}{\delta'} \rfloor + 1 \right) \delta', \ldots \right\} \cap (0, R(\sqrt{2}+\gamma)]$
13: ▷ The smallest element in W is $\Theta(1)$ by the setting of parameters. See 3.5.2
14: **for** $x'' \in W$ **do** ▷ Enumerate over potential target distances
15: **if** $x'' > R(\sqrt{2}+\gamma)$ **continue**
16: Choose η such that $\frac{F(\eta)}{G(x''/R,\eta)} = \left(\frac{1}{\mu}\right)^{\frac{1}{T}}$
17: ▷ Choose η such that a query explores $\left(\frac{1}{\mu}\right)^{\frac{1}{T}}$ children in expectation
18: **for** $i = 1, \ldots, \left\lceil \frac{100}{G(x''/R,\eta)} \right\rceil$ **do**
19: Sample a Gaussian vector $g \sim N(0,1)^d$
20: $P' \leftarrow \{\mathbf{p} \in P : \langle \frac{\mathbf{p}.new - o}{R}, g \rangle \geq \eta\}$
21: ▷ $\mathbf{p}.new$ is the rounded $\mathbf{p}$ to the surface of the sphere (see line 10 of Algorithm 9)
22: **if** $P' \neq \emptyset$ **then**
23: Add a node v' as child of w in $\mathcal{T}$
24: $v'.P \leftarrow P',\ v'.level \leftarrow v.level + 1,\ v'.g \leftarrow g,\ v'.r \leftarrow R,\ v'.o \leftarrow o,\ v'.x \leftarrow x''$
25: $v'.\eta \leftarrow \eta$
26: **if** $v'.level \neq J$ **then**
27: ▷ Stop whenever the level becomes J (see Section 3.5.2 for the value of J.)
28: SPHERICAL-LSH(v', x, μ) ▷ Recurse unless budget has been exhausted

Algorithm 13 is the standard (similar to Section 3.4) outer algorithm and is presented in Appendix C.3. It simply calls PREPROCESS (Algorithm 7) presented in this section. The following lemma bounds the space complexity of the preprocessing algorithm:

Lemma 3.5.2. *Under Assumption 3.5.1, the expected space consumption of the datastructure generated by* PREPROCESS-KDE$(\widetilde{P}, \mu)$ *(Algorithm 13) is bounded by*

$$\min\left\{ n \exp_\mu(0.173), \exp_\mu(1 + c + o(1)) \right\},$$

for small constant $c = 10^{-3}$.

Proof. First, we calculate the expected size of the data structure created by PREPROCESS$(\widetilde{P}, x, \mu)$ for any $x \in \{\delta_x, 2\delta_x, \ldots\} \cap (0, \sqrt{2})$ (see line 6 in Algorithm 13). Note that the expected size of the sampled dataset is

$$\mathbb{E}[|\widetilde{P}|] \leq \min\left\{\exp_\mu\left(1-\frac{x^2}{2}\right), n\right\}.$$

Since PSEUDORANDOMIFY does not duplicate points, every point in the dataset is duplicated (due to their presence in different spherical caps) at most

$$\exp_\mu\left(\frac{1}{T}\right)\cdot|W| = \exp_\mu\left(\frac{1}{T}\right)\cdot\exp\left((\log\log n)^{O(1)}\right)$$

times in expectation each time we increase the level. So, in total every point is duplicated at most

$$\exp_\mu\left(\frac{J}{T}\right)\cdot|W|^J = \exp_\mu\left(\frac{J}{T}\right)\cdot|W|^J$$

in expectation. Indeed, in every level we enumerate over at most $|W| = \exp((\log\log n)^{O(1)})$ possibilities for x'', amounting to at most a factor of $|W|^J = \exp\left((\log\log n)^{O(1)}\cdot O(\sqrt{\log n})\right) = n^{o(1)}$ duplication due to the termination condition in line 27 of Algorithm 8. Finally, PREPROCESS itself hashes every point $100\exp_\mu(\alpha) \leq 100\exp_\mu\left(10^{-4}\right)$ times (see line 6 and line 5 of Algorithm 7). Putting these bounds together yields that the space consumption of PREPROCESS$(\widetilde{P}, x, \mu)$ is at most

$$\min\left\{\exp_\mu\left(1-\frac{x^2}{2}\right), n\right\}\cdot\exp_\mu\left(10^{-4}+\min\left\{\alpha^*, \frac{x^2}{2}\left(1-\frac{x^2}{2}\right)+10^{-4}\right\}+o(1)\right)$$

in expectation, where $c := 10^{-4}$. Now, note that we also repeat this procedure $\left(\frac{1}{\mu}\right)^{4\delta_x+o(1)}$ times (see Algorithm 13), which results in the following bound on the total space consumption

$$\begin{aligned}&\max_{x\in(0,\sqrt{2})}\left(\min\left\{\exp_\mu\left(1-\frac{x^2}{2}\right), n\right\}\cdot\exp_\mu\left(10^{-4}+\min\left\{\alpha^*, \frac{x^2}{2}\left(1-\frac{x^2}{2}\right)+10^{-4}\right\}+4\delta_x+o(1)\right)\right)\\ &\leq \min\left\{n\exp_\mu(0.173), \exp_\mu(1+c+o(1))\right\},\end{aligned}$$

for $c = 10^{-3}$. □

Finally, we introduce the procedure PSEUDORANDOMIFY (Algorithm 9 below) used in SPHERICAL-LSH. This procedure is quite similar to the corresponding primitive in [13] and is guaranteed to output pseudo-random spheres with parameters τ and γ (See Definition 3.3.3).

Algorithm 9 PseudoRandomify

1: **procedure** PSEUDORANDOMIFY(v, γ)
2: $\delta \leftarrow \exp(-(\log\log n)^C)$
3: $R_{min} \leftarrow$ sufficiently small constant larger than Δ and δ_x (see Section 3.5.2)
4: $P \leftarrow v.P$ ▷ Dataset of node v
5: $R \leftarrow v.r$ ▷ Radius of sphere of node v
6: $o \leftarrow v.o$ ▷ Center of sphere of node v
7: $\tau \leftarrow \frac{1}{10}$
8: **if** $R < R_{min}$ **return**
9: **for** $\mathbf{p} \in P$ **do**
10: $\mathbf{p}.new \leftarrow o + \delta\lceil\frac{||\mathbf{p}-o||}{\delta}\rceil \cdot \frac{\mathbf{p}-o}{||\mathbf{p}-o||}$ ▷ $\mathbf{p}$ represents the initial coordinates of point $\mathbf{p}$
11: $V \leftarrow \emptyset$
12: **for** $i \leftarrow 1 \ldots \lceil\frac{R}{\delta}\rceil$ **do** ▷ Process all resulting spheres
13: $\widetilde{P} \leftarrow \{\mathbf{p} \in P : ||\mathbf{p}.new - o|| = \delta i\}$
14: **if** $\widetilde{P} \neq \emptyset$ **then**
15: $\hat{R} \leftarrow (\sqrt{2}-\gamma)R$
16: $m \leftarrow |\widetilde{P}|$
17: $m' \leftarrow 0$
18: **while** $m' \leq \frac{m}{2}$ **do**
19: $m \leftarrow |\widetilde{P}|$
20: **while** $\exists \hat{o} \in \partial B(o, \delta i) : |B(\hat{o}, \hat{R}) \cap \widetilde{P}| \geq \frac{1}{2} \cdot \tau \cdot m$ **do**
21: ▷ Using **rounded** $p.new$ coordinates (see line 10) in line above
22: $P' \leftarrow \widetilde{P} \cap B(\hat{o}, \hat{R})$
23: $B(o', R') \leftarrow \text{SEB}(P')$ ▷ SEB=smallest enclosing ball
24: Create a node tmp
25: $tmp.P \leftarrow P',\ tmp.level \leftarrow v.level,\ tmp.g \leftarrow 0,\ tmp.r \leftarrow R',\ tmp.o \leftarrow o'$
26: $V \leftarrow V \cup \text{PSEUDORANDOMIFY}(tmp, \gamma)$
27: $\widetilde{P} \leftarrow \widetilde{P} \setminus B(\hat{o}, \hat{R})$
28: $m' \leftarrow |\widetilde{P}|$
29: Create a node w
30: $w.P \leftarrow \widetilde{P},\ w.level \leftarrow v.level,\ w.g \leftarrow 0,\ w.r \leftarrow \delta i, w.o \leftarrow o$
31: $V \leftarrow V \cup \{w\}$
32: **return** V

3.5.3 Query procedure

We now present our query procedure (Algorithm 10 below). The procedure simply traverses the recursion tree $\mathcal{T}$ from the root, exploring leaves that the query is mapped to according to line 29. Since every node u of the tree $\mathcal{T}$ corresponds to a pseudorandom dataset $u.P$ residing (essentially) on a sphere of radius $u.r$ centered at $u.o$, the query is projected onto the sphere, after which one recursively explores the children of u in $\mathcal{T}$ whose Gaussian vectors (see line 29) are sufficiently correlated with the projected query. One notable feature in comparison to the corresponding procedure in [13] is the follows. Note that

the procedure of [13] recurses on a sphere even if the intersection of a sphere of radius x around the query (i.e. the range in which we would like to report points) barely touches the sphere that the dataset resides on. Our data structure, however, uses an increased search range $x + \Delta$ (see Figure 3.7), which results in somewhat higher runtime, but allows one to only recurse when the extended search range has nontrivial overlap with the sphere in question – see lower bound on x' in line 12 of Algorithm 8. This additive Δ technique, can also be used to simplify the technical proofs of [13], by not allowing their algorithm to recurse on tiny spheres at distance roughly x (i.e., when the distance x barely touches the sphere). The reason is that all the points on these small spheres has distance at most $x + 2R_{min}$ from the query, and we have small number of such points in expectation, by sub-sampling and density constraints.

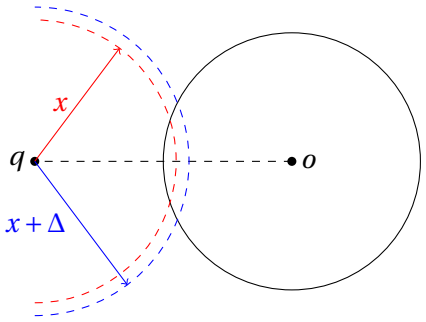


Figure 3.7: An (exaggerated) illustration of x and $x + \Delta$.

Algorithm 10 Query

1: **procedure** QUERY($\mathbf{q}, \mathcal{T}, x$)
2: $P_x \leftarrow \emptyset$
3: $\delta \leftarrow \exp(-(\log\log n)^C)$
4: $v \leftarrow$ root of $\mathcal{T}$
5: **if** $v.level = 0$ **then**
6: $K_1 = 100\left(\frac{1}{\mu}\right)^{\alpha}$ ▷ α is the constant from line 5 in Algorithm 7
7: **for** $j = 1, 2, \ldots, \lceil K_1 \rceil$ **do**
8: Locate $\mathbf{q}$ in h_j and $u \leftarrow$ the corresponding node in $\mathcal{T}$
9: **for** each w child of u **do**
10: $\mathcal{T}_w \leftarrow$ sub-tree of w and its descendants.
11: $P_x \leftarrow P_x \cup$ QUERY($\mathbf{q}, \mathcal{T}_w, x$)
return P_x
12: **else if** $\mathcal{T}$ is just one node, without any children **then**
13: **return** $v.P$
14: **else**
15: $o \leftarrow v.o$
16: $R \leftarrow v.r$
17: $R_2 \leftarrow ||\mathbf{q} - o||$
18: $x' \leftarrow$ PROJECT($x + \Delta, R_2, R$)
19: $x'' \leftarrow$ smallest element in the grid W (line 12 of Algorithm 8) which is not less than x'
20: **if** $x + \delta < |R - R_2|$ **then return**
21: ▷ Then no point from distance x can be on this sphere
22: **if** $\nexists u$ child of v, such that $u.x = x''$ **then**
23: **return** $v.P$
24: **for** each u child of v **do**
25: $\Delta \leftarrow 10^{-20}$
26: **if** $u.x = x''$ **then**
27: $g \leftarrow u.g$
28: $\eta \leftarrow u.\eta$
29: **if** $\langle g, \frac{\mathbf{q}-o}{||\mathbf{q}-o||} \rangle \geq \eta$ **then**
30: **for** each w child of u **do**
31: $\mathcal{T}_w \leftarrow$ sub-tree of w and its descendants.
32: $P_x \leftarrow P_x \cup$ QUERY($\mathbf{q}, \mathcal{T}_w, x$)
33: **return** P_x

Since the correctness analysis of this procedure is standard and similar to [13], we present it in Appendix C.3 (Lemma C.3.1), for completeness. Basically, we prove the query procedure outputs any given point within distance x with high constant probability.

3.6 Query time analysis

The main result of this section is the following lemma which bounds the expected query time of the algorithm.

Lemma 3.6.1. *The expected query time is bounded by* $O\left(\left(\frac{1}{\mu}\right)^{0.173+o(1)}\right)$.

Throughout this section we consider the setting where one is given a query $\mathbf{q} \in \mathbb{R}^d$ and a parameter $\mu \in (0,1]$ with the promise that

$$\mu^* \leq \mu, \tag{3.24}$$

where

$$\mu^* = K(P, \mathbf{q})$$

is the true kernel density value. We assume that $\mu^* = n^{-\Theta(1)}$, since this is the interesting regime for this problem. For $\mu^* = n^{-\omega(1)}$ under the Orthogonal Vectors Conjecture (e.g. [109]), the problem cannot be solved faster than $n^{1-o(1)}$ using space $n^{2-o(1)}$ [9], and for larger values $\mu^* = n^{-o(1)}$ random sampling solves the problem in $n^{o(1)}/\epsilon^2$ time and space.

Densities of balls around query. Upper bounds on the number of points at various distances from the query point in dataset (i.e., densities of balls around the query) play a central part in our analysis. The core of our query time bound amounts to tracking the evolution of such densities in the recursion tree $\mathcal{T}$. In order to analyze the evolution of these upper bounds we let, for a query $\mathbf{q} \in \mathbb{R}^d$ (which we consider fixed throughout this section) and any $x \in (0, \sqrt{2})$ let

$$D_x(\mathbf{q}) := \{||\mathbf{p}-\mathbf{q}|| : \mathbf{p} \in P, ||\mathbf{p}-\mathbf{q}|| \geq x + 1.5\Delta\}, \tag{3.25}$$

denote the set of possible distances from the query to the points in the dataset which are further that $x + 1.5\Delta$ from the query. When there is no ambiguity we drop q and x and we simply call it D. For any $y \in D$ we let

$$P_y(\mathbf{q}) := \{\mathbf{p} \in P : ||\mathbf{p}-\mathbf{q}|| \leq y\} \tag{3.26}$$

be the set of points at distance y from $\mathbf{q}$. Since for every $y > 0$

$$\begin{aligned} \mu^* = K(P, \mathbf{q}) &= \frac{1}{n} \sum_{\mathbf{p} \in P} \mu^{||\mathbf{p}-\mathbf{q}||_2^2/2} \\ &\geq \frac{\mu^{y^2/2}}{n} |P_y(\mathbf{q})| \end{aligned}$$

we get

$$|P_y(\mathbf{q})| \leq n\mu^* \cdot \left(\frac{1}{\mu}\right)^{\frac{y^2}{2}} \leq n \cdot \left(\frac{1}{\mu}\right)^{\frac{y^2}{2}-1},$$

since $\mu^* \leq 4\mu$ by assumption.

Densities in the subsampled dataset. Fix $x \in (0, \sqrt{2})$, and recall that $\widetilde{P}$ contains every point in P independently with probability $\frac{1}{n} \cdot \left(\frac{1}{\mu}\right)^{1-\frac{x^2}{2}}$. Note that for every y the expected number of points at distance at most y from query $\mathbf{q}$ that are included in $\widetilde{P}$ is upper bounded by

$$\min\left\{ n \cdot \left(\frac{1}{\mu}\right)^{\frac{y^2}{2}-1}, n \right\} \cdot \frac{1}{n} \cdot \left(\frac{1}{\mu}\right)^{1-\frac{x^2}{2}} \leq \min\left\{ \exp_\mu\left(\frac{y^2-x^2}{2}\right), \exp_\mu\left(\frac{1-x^2}{2}\right) \right\}, \tag{3.27}$$

and Figure 3.8 illustrates this.

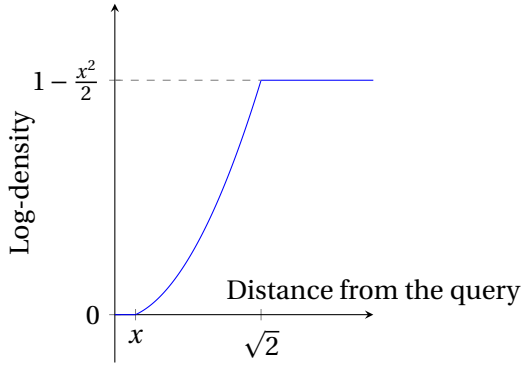


Figure 3.8: Upper-bound on log-densities after sub-sampling.

Our main goal is to track the progress of the query $\mathbf{q}$ and any $\mathbf{p}$, for which we have $||\mathbf{p}-\mathbf{q}||_2 \leq x$, that was included in the set $\widetilde{P}$, and exploit the upper bounds (3.27) on the number of points at various distances from $\mathbf{q}$ in $\mathbf{q}$'s 'hash bucket' to show that the process quickly converges to a constant size data set at a leaf of $\mathcal{T}$. It is not hard to see (Lemma 3.6.4 below) that the number of nodes in $\mathcal{T}$ that the query explores is low. The main challenge is to show that the expected size of a leaf data set in $\mathcal{T}$ is small, since for that one needs to prove strong upper bounds on the number of points at various distances from the query in dataset that the query traverses on its path to a leaf in $\mathcal{T}$. We exploit two effects:

(Removal of points due to truncation) The PSEUDORANDOMIFY procedure, which is crucial to ensuring that spheres at nodes on $\mathcal{T}$ are pseudorandom, essentially acts as a trunction primitive on the density curve. See conditions **(2)** in Definition 3.6.3 below.

(Removal of points due to LSH) As the query explores the children of an LSH node $v \in \mathcal{T}$ the probability that a given point $\mathbf{p} \in v.P$ belongs to the same spherical cap as $\mathbf{q}$ depends on the distance between $\mathbf{p}$ and $\mathbf{q}$. This implies bounds on the evolution of the density of points at various distances y from $\mathbf{q}$ in the datasets that $\mathbf{q}$ explores on its path towards a leaf in $\mathcal{T}$. See conditions **(3)** in Definition 3.6.3 below.

The bulk of our analysis is devoted to understanding the worst case sequence of geometric configurations, i.e. spheres, that the query encounters on its path towards a leaf in $\mathcal{T}$.

3.6.1 Path geometries

We start by defining the path geometries in the recursion tree. Assume an invocation of PREPROCESS algorithm (Algorithm 7) and let $\mathcal{T}$ be the sub-tree that the query explores. Let

$$\mathcal{P} := (w_0, v_0, w_1, v_1, \ldots, w_J, v_J)$$

be any path from root to a LSH leaf at level J.

For any $j \in [J]$, suppose that given $x'' := v_j.x$ and $r := v_i.r$, we are interested in the distance from the query to the center of the sphere $(v_j.o)$. For simplicity of notation let $\widetilde{\ell} = ||\mathbf{q} - v_j.o||$. Recall that x'' is the *r*ounded value for $x' = \text{PROJECT}(x+\Delta, \widetilde{\ell}, r)$ (see lines 18 and 19 of Algorithm 10). However, this equation is a degree two polynomial in $\widetilde{\ell}$, so it has at most two solutions. For intuition, Figure 3.9 shows these two solutions with an example. The solutions to the equation correspond to the points that the dashed circle intersects with the dashed line, i.e., position of $\mathbf{q}$. Now, recall that x'' is the rounded x' (see line 19 of Algorithm 10). So, x' can change in a small interval. This corresponds to moving the center of the dashed circle over the red arc. This changes the position of intersections, however, they still belong to a relatively small interval (shown in blue in Figure 3.9), we denote this intervals by *left interval* and *right interval*. Now, given query $\mathbf{q}$, we check weather it corresponds to the left interval or the right interval, and based on that we set b_j to be 1 or 2, respectively. We also let ℓ be the distance of the leftmost point in the interval of the query, from the center of the sphere. And we call ℓ the distance induced by (x'', r) and $\mathbf{q}$. In appendix C.4 we formally argue this procedure.

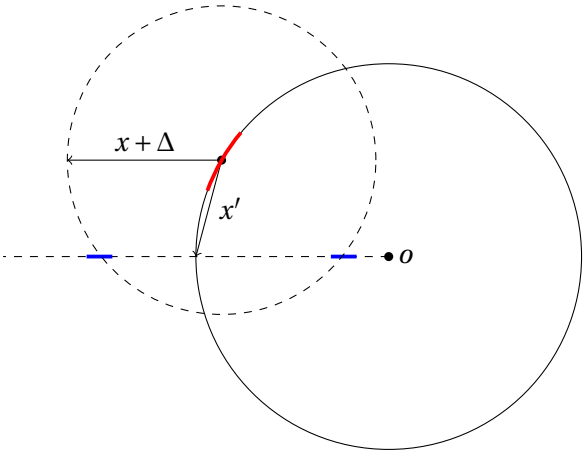


Figure 3.9: Geometric illustration of equation $x' = \text{PROJECT}(x+\Delta, \widetilde{\ell}, r)$ when we have access to an approximation of x' (red arc).

Definition 3.6.1 (Path geometry and induced distances)**.** *For any query* $\mathbf{q}$ *and tree* $\mathcal{T}$ *(as described above) for any root to leaf path*

$$\mathcal{P} = (w_0, v_0, w_1, v_1, \ldots, w_J, v_J),$$

we call

$$G(\mathcal{P}) := ((x''_1, r_1, b_1), \ldots, (x''_J, r_J, b_J))$$

the geometry of path $\mathcal{P}$ *where for all* $i \in [J]$,

1. $x_i'' := v_i.x$,

2. $r_i := v_i.r$,

3. b_i *is as described above (formally defined in Appendix C.4).*

Additionally, we call $L(\mathscr{P}) := (\ell_1, \ldots, \ell_J)$ *the induced distances of path* $\mathscr{P}$*, where for all* $i \in [J]$*,* ℓ_i *is induced by* (x_i'', r_i) *as explained above and formally defined in Appendix C.4.*

Definition 3.6.2 (Sphere geometries)**.** *For any query* $\mathbf{q}$ *and tree* $\mathcal{T}$ *(as described above) for any root to leaf path*

$$\mathscr{P} = (w_0, v_0, w_1, v_1, \ldots, w_J, v_J),$$

if the geometry of this path is defined as

$$G(\mathscr{P}) := ((x_1'', r_1, b_1), \ldots, (x_J'', r_J, b_J))$$

then for any $j \in [J]$ *we say that* w_j *and* v_j *has geometry* (x_j'', r_j, b_j)*.*

Recall from Definition 3.3.3 that the PSEUDORANOMIFY procedure (Algorithm 9) ensures that most of the points on any pseudorandom sphere w are nearly orthogonal to $\mathbf{q} - w.o$. We want to know, how the fact that a sphere is pseudorandom translates to densities. For the first step, we need to understand if a point on the sphere is almost orthogonal to the projection of the query on the sphere, then what the range of possible distances of these points from the query is. We define the $c := \sqrt{\ell^2 + r^2}$ which simplifies the notation. As Figure 3.10 suggests, we expect the orthogonal points to be at distance $\approx c$. The following claim formally argues how pseudorandomness of a sphere translates to a condition on the densities.

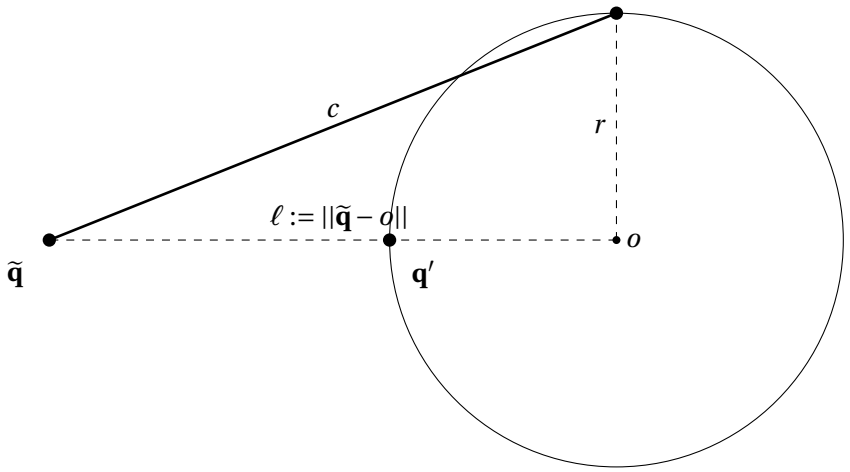


Figure 3.10: Illustration of the definition of c, the distance from the query to a 'typical' point on the sphere.

Claim 3.6.1 (Truncation claim)**.** *Given query* $\mathbf{q}$*, let* w *be a pseudo random sphere with geometry* (x'', r, b) *which induces distance* ℓ*. Let* $w.P$ *be the set of points on this sphere, i.e., for any* $\mathbf{p} \in w.P$*,* $\mathbf{p}.new$ *is on the sphere. For all* y *let* B_y *be the number of points at distance* y *from* $\mathbf{q}$ *in* $w.P$*. Then, the following conditions*

hold.

$$\sum_{y \le c - r\psi} B_y \le \frac{\tau}{1-2\tau} \cdot \sum_{y \in (c-r\psi, c+r\psi)} B_y,$$

and

$$\sum_{y \ge c + r\psi} B_y \le \frac{\tau}{1-2\tau} \cdot \sum_{y \in (c-r\psi, c+r\psi)} B_y,$$

where $\psi = \gamma^{1/3} + \delta'^{1/4} + \delta^{1/4}$ *(the same as in Claim C.5.1) and* $c := \sqrt{\ell^2 + r^2}$.

Proof. The proof is just a simple application of Claim C.5.1 to this sphere. □

Suppose that one has two points on the sphere at some distance from each other, we can use Lemma 3.3.1 and Lemma 3.3.2, to find collision probabilities under a spherical cap of size η. However, in general the query is not on the sphere, so we need to translate distance from $\mathbf{q}$ to any point $\mathbf{p}$ to distance from $\mathbf{q}'$ (projection of $\mathbf{q}$ on the sphere) to $\mathbf{p}$, using a function called PROJECT (formally defined in Definition 3.3.2 and its formula is given in Lemma 3.3.3). Also, there are some rounding steps, such as rounding the points to the sphere and rounding of the distance from the query to the center of the sphere (rounding of $\widetilde{\ell}$ to ℓ). Considering all these issues, the following claim illustrates the effect of spherical LSH on the points based on their distance from the query.

Claim 3.6.2 (Spherical LSH claim)**.** *Suppose that there is a sphere with geometry* (x'', r, b) *and induced distance* ℓ *(see Section 3.6.1 and Definition 3.6.1) for some* $x'' \in W$, $r \in \left[\left\lceil \frac{R_{max}}{\delta} \right\rceil\right]$ *and* $b \in \{1,2\}$. *Let* o *be the center of the sphere. Also, let* $\mathbf{p}$ *be a point such that* $y = ||\mathbf{p} - \mathbf{q}||$ *and* $\mathbf{p}.new$ *is on the sphere (see line 10 of Algorithm 9). Now, suppose that one generates a Gaussian vector* g *as in Algorithm 8. Then, we have*

$$\mathbb{P}_{g \sim N(0,1)^d}\left[\langle g, \frac{\mathbf{p}.new - o}{||\mathbf{p}.new - o||}\rangle \ge \eta | \langle g, \frac{\mathbf{q} - o}{||\mathbf{q} - o||}\rangle \ge \eta\right] \le \exp_\mu\left(-\frac{4(r/x')^2 - 1}{4(r/y')^2 - 1} \cdot \frac{1}{T}\right).$$

where

- η *is such that* $\frac{F(\eta)}{G(x''/r, \eta)} = \left(\frac{1}{\mu}\right)^{\frac{1}{T}}$ *(see line 16 of Algorithm 8).*
- $x' :=$ PROJECT$(x + \Delta, \ell, r)$.
- $y' :=$ PROJECT$(y - \Delta/2, \ell, r)$.

Proof. Let o be the center of the sphere. Let $\widetilde{\ell} := ||\mathbf{q} - o||$. Recall by the discussion in Section 3.6.1 and Definition 3.6.1 that any sphere geometry (x'', r, b) induces a distance ℓ. Now, suppose that we move the query in the direction of the vector from o to $\mathbf{q}$, such that for the new point $\widetilde{\mathbf{q}}$, we get $||\widetilde{\mathbf{q}} - o|| = \ell$. Now, one should note that the geometry of the sphere with respect to $\mathbf{q}$ and $\widetilde{\mathbf{q}}$ is the same. Also, the projections of $\mathbf{q}$ and $\widetilde{\mathbf{q}}$ on the sphere are identical. Also, for point $\mathbf{p}$ at distance y from $\mathbf{q}$, by the triangle inequality for

$(\mathbf{q}, \widetilde{\mathbf{q}}, \mathbf{p})$, since $\widetilde{\ell} \in [\ell - \delta'^{1/3}, \ell]$ we get

$$||\mathbf{p} - \widetilde{\mathbf{q}}|| \in [y - \delta'^{1/3}, y + \delta'^{1/3}]. \tag{3.28}$$

Now, if we let point $\mathbf{q}'$ be the projection of $\widetilde{\mathbf{q}}$ on the sphere, and let $\mathbf{p}.new$ be the rounded $\mathbf{p}$ on the sphere, then $||\mathbf{p}.new - \widetilde{\mathbf{q}}|| \in [y - \delta - \delta'^{1/3}, y + \delta + \delta'^{1/3}]$, which implies

$$y'' := ||\mathbf{q}' - \mathbf{p}.new|| \in [\textsc{Project}(y - \delta - \delta'^{1/3}, \ell, r), \textsc{Project}(y + \delta + \delta'^{1/3}, \ell, r)].$$

Note that with this definition of y'' one has

$$\mathbb{P}_{g \sim N(0,1)^d} \left[\langle g, \frac{\mathbf{p}.new - o}{||\mathbf{p}.new - o||} \rangle \geq \eta | \langle g, \frac{\mathbf{q} - o}{||\mathbf{q} - o||} \rangle \geq \eta \right] = \frac{G(y''/r, \eta)}{F(\eta)}. \tag{3.29}$$

Now, by invoking Claim C.5.2, **(b)**

$$\mathbb{P}_{g \sim N(0,1)^d} \left[\langle g, \frac{\mathbf{p}.new - o}{||\mathbf{p}.new - o||} \rangle \geq \eta | \langle g, \frac{\mathbf{q} - o}{||\mathbf{q} - o||} \rangle \geq \eta \right] \leq \exp_\mu \left(- \frac{4(r_j/x')^2 - 1}{4(r_j/y')^2 - 1} \cdot \frac{1}{T} \right) \tag{3.30}$$

Now, we verify the preconditions of Claim C.5.2, **(b)**. Condition **(p1)** of Claim C.5.2 is satisfied by setting of δ as $\delta + \delta'^{1/3}$.[12] Condition **(p2)** is satisfied by line 10 in Algorithm 8. Condition **(p3)** is satisfied by setting of Δ in line 25 of Algorithm 10. Finally, condition **(p4)** is satisfied due to line 15 in Algorithm 8 that ensures that a nontrivial data structure is only prepared for $x' \leq R(\sqrt{2} + \gamma)$, and no recursion is performed otherwise.

Conditioned on event $\mathscr{E}_{diam}$ (which ensures constant upper-bound on the radii of spheres, see the discussion in Section 3.5.1), $r = O(1)$. Thus, we can invoke part **(b)** of Claim C.5.2 applies and gives (3.30). □

In the following definition we summarize the effect of sub-sampling the dataset, the truncation rounds and the spherical LSH rounds on densities along the path.

[12] To be more clear, we set the δ of claim C.5.2 as $\delta + \delta'^{1/3}$ where δ and δ' are the parameters of the algorithm.

Definition 3.6.3 (Valid execution path)**.** *Let $R := (r_j)_{j=1}^J$ and $L := (\ell_j)_{j=1}^J$ for some positive values r_j's and ℓ_j's such that for all $j \in [J]$, $x + \delta \geq |\ell_j - r_j|$. Also let D be as defined in* (3.25)*. Then, for*

$$A := (a_{y,j}), \quad y \in D, j \in [J] \cup \{0\} \qquad \textit{(Intermediate densities)}$$
$$B := (b_{y,j}), \quad y \in D, j \in [J+1] \cup \{0\} \qquad \textit{(Truncated intermediate densities)}$$

(L, R, A, B) *is called a* valid execution path*, if the conditions below are satisfied. We define $\psi := \gamma^{1/3} + \delta'^{1/4} + \delta^{1/4}$ and $c_j := \sqrt{r_j^2 + \ell_j^2}$ for convenience.*

(1) Initial densities condition. *The $a_{y,0}$ and $b_{y,0}$ variables are upper-bounded by the initial expected densities in the sampled dataset: for all $y \in D$*

$$\sum_{y' \in [0,y] \cap D} a_{y',0} \leq \min\left\{\exp_\mu\left(\frac{y^2 - x^2}{2}\right), \exp_\mu\left(\frac{1 - x^2}{2}\right)\right\}$$

and

$$\sum_{y' \in [0,y] \cap D} b_{y',0} \leq \min\left\{\exp_\mu\left(\frac{y^2 - x^2}{2}\right), \exp_\mu\left(\frac{1 - x^2}{2}\right)\right\}$$

(2) Truncation conditions (effect of PSEUDORANDOMIFY). *For any $j \in [J]$, for all $y \in D \setminus [\ell_j - r_j, \ell_j + r_j]$ one has $b_{y,j} = 0$ (density is zero outside of the range corresponding to the j-th sphere on the path; condition* **(2a)***), for all $y \in D \cap [\ell_j - r_j, \ell_j + r_j]$ one has $b_{y,j} \leq a_{y,j-1}$ (removing points arbitrarily* **(2b)***) and*

$$\sum_{y \in [0, c_j - \psi r_j] \cap D} b_{y,j} \leq \frac{\tau}{1 - 2\tau} \cdot \sum_{y \in (c_j - \psi r_j, c_j + \psi r_j) \cap D} b_{y,j} \qquad \textit{(condition } \textbf{(2c)}\textit{)}$$

(3) LSH conditions. *For every $j \in [J]$ and all $y \in [\ell_j - r_j, \ell_j + r_j] \cap D$*

$$a_{y,j} \leq b_{y,j} \cdot \exp_\mu\left(-\frac{4\left(\frac{r_j}{x'}\right)^2 - 1}{4\left(\frac{r_j}{y'}\right)^2 - 1} \cdot \frac{1}{T}\right)$$

where $x' := \text{PROJECT}(x + \Delta, \ell_j, r_j)$ and $y' := \text{PROJECT}(y - \Delta/2, \ell_j, r_j)$. See Remark 3.6.1 below for a discussion about Δ factors.

(4) Terminal density condition. *For any y such that $a_{y,J}$ is defined, $b_{y,J+1} \leq a_{y,J}$.*

Remark 3.6.1. *Throughout the paper we need good bounds on the probability that a random spherical cap encompasses a data point* **p***, given that the spherical cap captures the projection of the query. The expression in condition* **(3)** *of Definition 3.6.3 is a convenient upper bound for this quantity when the distance from* **p** *to* **q** *is equal to y. Exact expressions for such collision probabilities are unstable with respect to perturbations of the point* **p** *when* **p** *is antipodal to* **q** *on the sphere, and because of this it is more convenient to work with upper bounds. Specifically, we upper bound this probability by imagining that the point is slightly closer (by $\Delta/2$) than the actual distance y, for a small positive constant Δ that affects our query time bounds. The advantage is that such probabilities are more stable under small perturbations of the data point* **p** *– see the proof of Claim C.5.2 for more details. One notes that the expression in condition* **(3)** *also depends on x. This*

is because we select spherical cap sizes based on x – see line 16 of Algorithm 8.

We introduce the notion of the length of an execution path (L, R, A, B).

Definition 3.6.4. *We define the length of an execution path (L, R, A, B) by Length$((L, R, A, B)) := |R| = J$.*

A special class of execution paths that we refer to as zero-distance monotone paths will be central to our analysis:

Definition 3.6.5. *(Zero-distance and monotone path) Let (L, R, A, B) be an execution path defined in Definition 3.6.3. If for $R = (r_j)_{j=1}^{J}$, r_j's are non-increasing in j, and $L = R$, then we say that (L, R, A, B) is a* zero-distance and monotone *execution path. When $L = R$, we usually drop L, and simply write (R, A, B).*

The following crucial lemma allows our LP based analysis of the query time:

Lemma 3.6.2. *(Reduction to zero-distance monotone execution paths) For every valid execution path (L, R, A, B) (see Definition 3.6.3), there exists a* zero-distance and monotone *valid execution path (R', A', B') (see Definition 3.6.5) such that $b'_{y,J+1} = b_{y,J+1}$ for all $y \in D$[13] and $|R'| = |R|$ (i.e., the length of the paths are equal).*

The proof of this lemma is given in Section 3.7.

Linear programming formulation

As we prove in Lemma 3.6.2, for any execution path there exists a zero-distance monotone path (see Definition 3.6.5) with the same length and the same final densities. This means that if we prove that for any zero-distance monotone path, the final densities are small, then this generalizes to all possible execution paths. So, from now on we only consider zero-distance monotone paths.

As mentioned before, we analyze the evolution of density of points at various distances. Instead of analyzing continuous densities, we define a new notion, called *discretized log-densities* (see Definition 3.6.7), for which we round densities to the discretized distances in a natural way, and for simplicity of calculations we take the log of these densities. These two steps allow us to analyze the evolution of densities over the course of time. More specifically, we define an LP (see (3.33)) such that any zero-distance monotone execution path with large enough final densities, imposes a feasible solution to the LP, with cost (almost) equal to the length of the execution path divided by T. Thus, if the length of the execution path is large, final densities cannot be too large (see Section 3.8 and Claim 3.8.6 for the formal statement), which means that we managed to reduce the densities to a small amount.

In section 3.8 we formally describe the procedure for constructing a feasible solution based on discretized densities.

We start by defining a convenient discretization of the distances on a valid execution path:

[13] We need the final condition to argue that we have the same number of points remaining at the end.

Definition 3.6.6 (x-centered grid Z_x)**.** *For every $x \in (0, R_{max})$ define the grid $Z_x = \{z_I, z_{I-1}, \ldots, z_0\}$ by letting $z_I = x$, letting $z_{I-i} := (1+\delta_z)^i \cdot z_I$ for all $i \in [I]$ and choosing the smallest integer I such that $z_0 \geq R_{max}\sqrt{2}$.*

Definition 3.6.7 (Discretized log-densities $f_{z_i,j}$)**.** *For any zero-distance monotone valid execution path (R, A, B) (as per Definition 3.6.3) with radii bounded by R_{max} and $J = |R|$, for all $j \in [J]$ let k_j be the index of the largest grid element which is not bigger than $r_j \cdot (\sqrt{2}+\psi)$, i.e.,*

$$r_j \cdot (\sqrt{2}+\psi) \in [z_{k_j}, z_{k_j-1}) \tag{3.31}$$

and for every integer $i \in \{k_j, \ldots, I\}$ define

$$f_{z_i,j} := \log_{1/\mu}\left(\sum_{y \in D \cap [z_{i+1}, z_{i-1})} b_{y,j}\right) \tag{3.32}$$

Note that the variables $b_{y,j}$ on the right hand side of (3.32) *are the $b_{y,j}$ variables of the execution path (R, A, B).*

Letting $Z := Z_x$ to simplify notation, we will consider I linear programs defined below in (3.33), enumerating over all $j^* \in [I]$, where we let $x' = x + \Delta$:

$$\begin{aligned}
\mathrm{LP}(x, j^*): \quad & \max_{\alpha \geq 0} \sum_{j=1}^{j^*-1} \alpha_j && (3.33)\\
& \forall y \in Z: g_{y,1} \leq \min\left\{\frac{y^2-x^2}{2}, 1-\frac{x^2}{2}\right\} && \text{Density constraints}\\
& \text{for all } j < j^*, y \in Z, y < z_j: && \\
& \quad g_{y,j} \leq g_{z_j,j} && \text{Truncation}\\
& \quad g_{y,j+1} \leq g_{y,j} - \frac{2\left(z_j/x\right)^2-1}{2\left(z_j/y\right)^2-1} \cdot \alpha_j && \text{Spherical LSH}\\
& \quad g_{z_{j^*},j^*} \geq 0 && \text{Non-empty range constraint}
\end{aligned}$$

Intuitively, LP (3.33) captures the evolution of the density of points at different distances from the query throughout the hashing process. Our main technical claim connecting the LP (3.33) and execution paths in the query process is Claim 3.8.6 in Section 3.8.

3.6.2 Upper-bounding the expected number of points examined by the query

In this section we bound the expected number of points that the query examines in the query procedure. Let $\mathscr{T}$ be the tree that the query traverses. Note that the query only examines the points that it sees in the leaves that it visits. One should note that some leaves (which are LSH nodes for this case) in the tree have level J (see line 27 of Algorithm 8). However, they are other leaves in tree $\mathscr{T}$, due to two cases:

1. Path termination due to $x'' > R(\sqrt{2}+\gamma)$. This case happens when query $\mathbf{q}$ is such that it needs to recover points at distance x'' on the sphere, but this distance corresponds to points beyond orthogonal. Note that in the preprocessing phase we did not prepare any child with this x'' (see line 15 in Algorithm 8), so the query will stop at this node and scan the points (see line 22 of Algorithm 10). Roughly speaking, since we only expect $O(1)$ number of points at distance x, and since the number of points on the sphere is dominated by the number of points in the orthogonal band, then we expect to see small number of points on this sphere. We formally prove this in Claim 3.6.3.

2. Path termination due to small sphere radius. This simple case corresponds to the cases when PSEUDORANDOMIFY does not process a ball further due to line 8 of Algorithm 9 or SPHERICALLSH does not partition the dataset further due to line 9 of Algorithm 8. Note that in that case the entire ball is necessarily at distance at most $x + 2R_{min}$, and hence the total number of points in the ball is small. We formally argue and prove this in Claim 3.6.3.

Claim 3.6.3. *For any tree $\mathcal{T}$ that the query $\mathbf{q}$ explores, the expected total number of points in the leaves with level less than J is bounded by*

$$\left(\frac{1}{\mu}\right)^{\alpha+\alpha^*+c},$$

for $c = 10^{-4}$.

Proof. We investigate the two cases mentioned above separately:

Path termination due to $x'' > R(\sqrt{2}+\gamma)$. First, suppose that the exploration process terminates at node $u \in \mathcal{T}$ because of line 15 in Algorithm 8 . In that case one has by invoking Claim 3.3.1 for two diametral points on the sphere, since the current dataset $u.P$ is pseudorandom as per Definition 3.3.3 and $\tau = 1/10$,

$$\left|\left\{\mathbf{p} \in u.P : ||\mathbf{p}-\mathbf{q}'|| \in \left(R(\sqrt{2}-\gamma), R(\sqrt{2}+\gamma)\right)\right\}\right| = \Omega\left(|u.P|\right).$$

Note that the expected number of points at distance at most $R(\sqrt{2}+\gamma)$ from the query is upper-bounded by the expected number of points at distance at most $x+\Delta+\delta'$, since $x'' > R(\sqrt{2}+\gamma)$ and by rounding of x' to x'' (see line 19 in Algorithm 10). So, after sub-sampling the data set and using the density constraints, we have at most

$$\begin{aligned}
\frac{1}{n}\cdot\left(\frac{1}{\mu}\right)^{1-x^2/2}\cdot 4n\cdot\mu^{1-\frac{(x+\Delta+\delta')^2}{2}} &= 4\exp_\mu\left(\frac{\left(x+\Delta+\delta'\right)^2}{2}-\frac{x^2}{2}\right)\\
&\le 4\exp_\mu\left(\frac{(x+2\Delta)^2}{2}-\frac{x^2}{2}\right)\\
&\le 4\exp_\mu\left(\frac{1}{2}\cdot(4\Delta x+4\Delta^2)\right)\\
&\le \exp_\mu(5\Delta) \qquad \text{Since } x\le\sqrt{2} \text{ and } \Delta = 10^{-20}
\end{aligned}$$

points.

Path termination due to small sphere radius. As we discussed above for this case, the entire ball is necessarily at distance at most $x+2R_{min}$, since this sphere passed the condition in line 20 of Algorithm 10, and hence on expectation the total number of points in this ball is bounded by

$$\frac{1}{n}\cdot\left(\frac{1}{\mu}\right)^{1-x^2/2}\cdot n\cdot\mu^{1-(x+2R_{min})^2/2}\leq \exp_\mu(4R_{min})$$

where the last line is by our choice of parameters, and since $x\leq\sqrt{2}$.

Also, by Lemma 3.6.4 we know that the query explores at most $\left(\frac{1}{\mu}\right)^{\alpha^*+\alpha+o(1)}$ leaves. Now, by setting of parameters, the claim holds. □

Lemma 3.6.3. *Under Assumption 3.5.1, there exists an event $\mathcal{E}$ that depends on the choice of the hash function in* PREPROCESS *only and occurs with probability at least $1-(1/\mu)^{-4}$ such that conditioned on $\mathcal{E}$, the following holds. The query examines at most*

$$\left(\frac{1}{\mu}\right)^{0.173} \tag{3.34}$$

number of points in expectation.

Proof. First, we just calculate the expected size of the data set examined by the query in invocations of QUERY (Algorithm 10), and then we bound the expected total number of points of QUERY-KDE (Algorithm 14). Note that the goal is to prove an upper-bound on the expected number of points that the query examines.

Consider an invocation PREPROCESS and let $\mathcal{T}$ be the sub-tree of the recursion tree that the query explores. Now, we define processes on this tree that output a subset of leaves of this tree. Suppose that

$$\mathcal{H}=W\times\left\lceil\left\lceil\frac{R_{\max}}{\delta}\right\rceil\right\rceil\times\{1,2\}$$

And let J be the maximum number of times that we applied spherical LSH. Let $\mathbf{q}$ be the query. Let $M:=|\mathcal{H}^J|$ and enumerate elements in $\mathcal{H}^J$. For any leaf in $\mathcal{T}$ if one looks at the path to the root from this leaf, this corresponds to one element in $\mathcal{H}^J$ (See the discussion in Section 3.6.1 and Definition 3.6.1). For i'th element of $\mathcal{H}^J$, $h_i=(h_i(j))_{j=1}^J$, the procedure $\mathcal{P}_i(\mathcal{T})$ outputs set E_i, which is the set of output(s) of SAMPLE($\mathcal{T},h_i,0$).[14] Note that Algorithm 11 outputs a set of leaves in the tree.

[14]Also, for the purpose of consistency define $h_i(0)=(0,0,0)$ and let $h_i\leftarrow(h_i(j))_{j=0}^J$ and assume that every Andoni-Indyk LSH bucket is consistent with $h_i(0)$.

Algorithm 11

1: **procedure** SAMPLE($\mathcal{T}, h_i, k$)
2: $v \leftarrow$ a uniformly random child of the root of $\mathcal{T}$ which is consistent with $h_i(k)$.
3: **if** $k = J$ **then**
4: Return v
5: **for** all w in the set of the childern of v **do**
6: **if** w is consistant with $h_i(k)$ **then**
7: $\mathcal{T}' \leftarrow$ the sub-tree of tree where the root is w.
8: SAMPLE($\mathcal{T}', h, k+1$).

Also, for any pseudo-random node on the tree that the query visits, since $\mu = n^{-\Omega(1)}$ by assumption, using a simple Chernoff bound argument, we have that it explores at most

$$m := O(1)\left(\frac{1}{\mu}\right)^{\frac{1}{T}}$$

children of this node, with high probability.

Let V be the set of leaves in $\mathcal{T}$, with level J. Partition V into $V_1, \ldots, V_M$, such that for all $i \in [M]$, the leaves in V_i admit the geometry defined by h_i.

Claim 3.6.4. *For any $u \in U_i$ we have the following*

$$\mathbb{P}[u \in E_i | \mathcal{T}] \geq \left(\frac{1}{m}\right)^J \left(\frac{1}{100\left(\frac{1}{\mu}\right)^{\alpha}}\right).$$

Proof. There is exactly one path from root to u. So, $u \in E_i$ if in all choices in line 2 of Algorithm 11, the algorithm chooses the correct child. This happens with probability at least $\left(\frac{1}{m}\right)^J \left(\frac{1}{100\left(\frac{1}{\mu}\right)^{\alpha}}\right)$. To be more clear, with probability $\left(\frac{1}{100\left(\frac{1}{\mu}\right)^{\alpha}}\right)$ the correct child of the root is chosen, and the other term correspond to the success probability in J steps. □

Now, we have the following:

$$\begin{aligned}
\sum_{i\in[M]} \mathbb{E}\left[\sum_{v\in E_i} |v.P|\,|\mathcal{T}\right] &= \sum_{i\in[M]} \mathbb{E}\left[\sum_{u\in V_i} \mathbb{I}\{u \in E_i\}|u.P|\,|\mathcal{T}\right] \\
&= \sum_{i\in[M]} \sum_{u\in V_i} \mathbb{P}[u \in E_i|\mathcal{T}] \cdot |u.P| \\
&\geq \left(\frac{1}{m}\right)^J \sum_{i\in[M]} \sum_{u\in V_i} |u.P| \\
&= \left(\frac{1}{m}\right)^J \sum_{u\in V} |u.P| \qquad (3.35)
\end{aligned}$$

where expectations are over the random choices of line 2 of Algorithm 11.

Let V' be the leaves with level $\neq J$. Note that $\sum_{u\in V}|u.P| + \sum_{u\in V'}|u.P|$ is equal to the number of points that the query examines in the leaves of $\mathcal{T}$. Note that Claim 3.6.3 proves that

$$\mathbb{E}_{\mathcal{T}}\left[\sum_{u\in V'}|u.P|\right] \leq \left(\frac{1}{\mu}\right)^{\alpha+\alpha^*+0.0001} \tag{3.36}$$

Now, we need to take expectation over the tree $\mathcal{T}$. From now on, the goal is to prove an upper-bound on

$$\mathbb{E}_{\mathcal{T}}\left[\mathbb{E}\left[\sum_{v\in E_i}|v.P|\,|\mathcal{T}\right]\right]$$

where the outer expectation is over the randomness of trees, and the inner expectation is over the randomness of choices in line 2 of Algorithm 11.

For any $\mathcal{T}$, define $W^{(0,\mathcal{T})}$, as the root of T. For all $j \in [J]\cup\{0\}$ let $V^{(j,\mathcal{T})}$ be the nodes in the tree selected by line 2 of Algorithm 11, when $k = j$. Also, for all $j \in [J]$ let $W^{(j,\mathcal{T})}$ be the children of $V^{(j-1,\mathcal{T})}$ which are consistent with $h_i(j)$, .i.e., nodes satisfying the condition in line 6 of Algorithm 11 when $k = j$. We drop superscripts for the tree, when it is clear from the context.

For all $j \in [J]\cup\{0\}$, $A_{y,j}$ denote the number of points at distance y for all $y \geq x + 1.5\Delta$ from the query in $\cup_{u\in V^{(j)}} u.P$. And similarly, for all $j \in [J]\cup\{0\}$ define $B_{y,j}$ as the number of points at distance y from the query in $\cup_{u\in W^{(j)}} u.P$.

Also, let $L = (\ell_j)_{j=1}^{J}$ be the distances induced by the geometry h_i. Now, define $x'_j := \text{Project}(x+\Delta, \ell_j, r_j)$ and $y'_j = \text{Project}(y-\Delta/2, \ell_j, r_j)$. Now, Claim 3.6.2 implies that for all $j \in [J]$

$$\mathbb{E}\left[A_{y,j}|B_{y,j}, \mathcal{T}_{<j}\right] \leq p_{y,j}\cdot B_{y,j}, \tag{3.37}$$

where the expectation is over the randomness of the tree and the random choice of line 2 of Algorithm 11, and

$$p_{y,j} := \exp_{\mu}\left(-\frac{4(r_j/x'_j)^2-1}{4(r_j/y'_j)^2-1}\cdot\frac{1}{T}\right). \tag{3.38}$$

On the other hand, since $B_{y,j}$ variables correspond to pseudo-random spheres, using Claim 3.6.1 they should satisfy the following:

$$\sum_{y\leq c_j-\psi R_j} B_{y,j} \leq \frac{\tau}{1-2\tau}\cdot\sum_{y\in(c_j-\psi R_j, c_j+\psi R_j)} B_{y,j}, \tag{3.39}$$

and

$$\sum_{y\geq c_j+\psi R_j} B_{y,j} \leq \frac{\tau}{1-2\tau}\cdot\sum_{y\in(c_j-\psi R_j, c_j+\psi R_j)} B_{y,j}. \tag{3.40}$$

Also, since $\cup_{u\in V^{(j)}} u.P \subseteq \cup_{u\in W^{(j)}} u.P$, then $B_{y,j} \leq A_{y,j-1}$. At this point, define Now, for all $j \in [J]$ define

$$\widetilde{B}_{y,j} := \mathbb{E}\left[B_{y,j}\right]$$

and

$$\widetilde{A}_{y,j} := \widetilde{B}_{y,j} \cdot p_{y,j} \tag{3.41}$$

and define

$$\widetilde{B}_{y,J+1} := \widetilde{A}_{y,J}. \tag{3.42}$$

Therefore, if $A := (\widetilde{A}_{y,j})_{j=1}^{J}$ and $B := (\widetilde{B}_{y,j})_{j=1}^{J+1}$ and L is the ordered set of distances induced by the path geometry h_i (see Section 3.6.1 and Definition 3.6.1) and R is the set of radii of the spheres, then we can argue that (L, R, A, B) is a valid execution path by Definition 3.6.3. Checking the conditions of Definition 3.6.3:

- Initial conditions: They are satisfied by the expectation of sub-sampling (see (3.27)), i.e.,

$$\sum_{y' \in [0,y] \cup D} \widetilde{A}_{y',0} \le \min\left\{\exp_\mu\left(\frac{y^2-x^2}{2}\right), \exp_\mu\left(\frac{1-x^2}{2}\right)\right\}$$

  and

$$\sum_{y' \in [0,y] \cup D} \widetilde{B}_{y',0} \le \min\left\{\exp_\mu\left(\frac{y^2-x^2}{2}\right), \exp_\mu\left(\frac{1-x^2}{2}\right)\right\}.$$

- Truncation conditions: **(2a)** is satisfied since if a point is on the sphere, its distance to the query can be in interval $[x + 1.5\Delta, \ell_j + r_j]$ which is a sub-interval of $[\ell_j - r_j, \ell_j + r_j]$, by the definition of induced distances and setting of parameters. **(2b)** holds, since the number of points in each distance is non-increasing from root to leaf. **(2c)** is satisfied by (3.39).

- LSH conditions: They are satisfied by (3.41) and the definition of $p_{y,j}$ in (3.38).

- Terminal density condition: It holds by (3.42).

we conclude that (L, R, A, B) is a valid execution path.

Now by Lemma 3.6.2 there exists a zero distance monotone execution path (R', A', B') such that $A' = (a'_{y,j})$, $B' = (b'_{y,j})$ and $b'_{y,J+1} = \widetilde{B}_{y,J+1}$. Let $f_{y,j}$'s be defined based on $b'_{y,j}$'s using Definition 3.6.7. More specifically, for every integer $i \in \{k_j, \ldots, I\}$ (see Definition 3.6.7 for the definition of k_j) define

$$f_{z_i,j} := \log_{1/\mu}\left(\sum_{y \in D \cap [z_{i+1}, z_{i-1})} b'_{y,j}\right) \tag{3.43}$$

Now, by Claim 3.8.6 and our setting of J (see Section 3.5.2), which ensures that $J > \frac{T}{1-10^{-4}}\text{OPT(LP)}$, for all $y \le z_{j^*-1}$ we have $f_{y,J+1} < 7\delta_z$ for $j^* = k_J + 1$. Now, we need to prove that this implies that $\sum_y \widetilde{A}_{y,J}$ is small:

Claim 3.6.5. *If for all $y \le z_{j^*-1}$ we have $f_{y,J+1} < 7\delta_z$ for $j^* = k_J + 1$, then we have the following bound*

$$\sum_y \widetilde{A}_{y,J} \le \exp_\mu(7\delta_z + o(1)).$$

The proof is deferred to Appendix C.5.

We just proved that for any fixed $i \in [M]$, $\sum_y \widetilde{A}_{y,J}$ (which bounds the expected number of points at distance $\ge x+1.5\Delta$ (see (3.25)) that the query examines in buckets with geometry h_i) is bounded by $\exp_\mu(7\delta_z + o(1))$. Moreover, recall that in this process we only considered points at distance $\ge x + 1.5\Delta$. We should also add the contribution of points at distance $< x + 1.5\Delta$. For this, just recall that after sub-sampling (even without considering any LSH effect on these points) in expectation we have

$$4\left(\frac{1}{\mu}\right)^{\frac{(x+1.5\Delta)^2 - x^2}{2}} \le \left(\frac{1}{\mu}\right)^{10\Delta}. \tag{3.44}$$

Now, in order to argue that the expected number of points examined by the query is bounded, we need to multiply by M, which results in the following bound

$$M \cdot \left(\exp_\mu(7\delta_z + o(1)) + \exp_\mu(10\Delta)\right) \tag{3.45}$$

which by the setting of parameters, combining with (3.35) and summing with (3.36), and considering the we call QUERY (Algorithm 10) at most $\left(\frac{1}{\mu}\right)^{4\delta_x + o(1)}$, gives the following bound on the expected number of points scanned by the query

$$\left(\frac{1}{\mu}\right)^{0.173}.$$

□

3.6.3 Proof of Lemma 3.6.1

Before we present the proof of Lemma 3.6.1, we need to show another auxiliary claim that helps us establish an upper bound on the expected number of leaves that a query explores, which helps upper bound the work done to reach a leaf (recall that Lemma 3.6.3 shows that the expected size of the dataset corresponding to a leaves of $\mathcal{T}$ that the query scans is bounded, so combining these two bounds will give us the final result).

Lemma 3.6.4. *For every $\mathbf{q} \in \mathbb{R}^d$, every $x > 0$, every $\mu \in (0,1)$ under Assumption 3.5.1, if $\mathcal{T}$ is the tree generated by* PREPROCESS(P, x, μ)*, then the expected number of leaves explored by a query q in a call to* QUERY$(\mathbf{q}, x, \mathcal{T})$ *(Algorithm 10) is bounded by* $\exp_\mu(\alpha^* + \alpha + o(1))$.

The proof is given in Appendix C.5. We will also need the following technical claim, which we also prove in Appendix C.5.

Claim 3.6.6. *For every $R \geq R_{min}$, for every $x' \in (\Delta, R(\sqrt{2}+\gamma))$ and sufficiently large η we have that $\frac{1}{G(x'/R,\eta)} = \left(\frac{F(\eta)}{G(x'/R,\eta)}\right)^{O(1/\Delta^2)}$.*

Proof of Lemma 3.6.1: By Lemma 3.6.4 the expected number of leaves that the query explores is bounded by

$$\left(\frac{1}{\mu}\right)^{\alpha^*+\alpha+o(1)} \tag{3.46}$$

The expected size of the dataset that the query scans is bounded by $\left(\frac{1}{\mu}\right)^{0.173}$ with high probability by Lemma 3.6.3. Now by an application of Markov's inequality to (3.46) we have that the query explores at most $\left(\frac{1}{\mu}\right)^{0.173+o(1)}$ leaves with high probability, and hence the total work is bounded by $(\frac{1}{\mu})^{0.173} \cdot n^{o(1)}$, as required. Finally, we bound the work done in line 18 of Algorithm 8. Indeed, recall that $x' < R(\sqrt{2}+\gamma)$ by line 15 of Algorithm 8, and at the same time by Claim 3.6.6 we have

$$\frac{1}{G(x'/R,\eta)} = \left(\frac{F(\eta)}{G(x'/R,\eta)}\right)^{O(1/\Delta^2)}.$$

Equipped with this observation, we can now finish the proof. We get using the choice of T in line 16 of Algorithm 8

$$\frac{100}{G(x'/R,\eta)} = 100 \cdot \left(\frac{F(\eta)}{G(x'/R,\eta)}\right)^{O(1/\Delta^2)} = 100 \cdot (1/\mu)^{O(1/(\Delta^2 \cdot T))} = n^{o(1)}$$

by choice of $\Delta = \Omega(1)$ and $T = \sqrt{\log n}$ in line 3 of Algorithm 8. This completes the proof. □

3.7 Reduction to zero-distance monotone execution paths

In this section, we prove Lemma 3.6.2, which proves that for any valid execution path, there exists a zero-distance valid execution path such that the final densities are identical and both have the same length. First, we state the following claims, and then assuming these claims, we prove Lemma 3.6.2. Then, we present the proof of these claims.

Claim 3.7.1 (Reduction to zero distance paths)**.** *For any L, R, A and B such that (L,R,A,B) is a valid execution path (see Definition 3.6.3), there exists R' and A' such that (R',R',A',B) is a valid execution path for some A'.*

Claim 3.7.2 (Local improvement towards monotonicity)**.** *For every valid zero-distance execution path (R,A,B), if for some $i \in [J-1]$ one has $r_i \leq r_{i+1}$, then for $R' := (r_1,\ldots,r_{i-1},r_{i+1},r_{i+1},\ldots,r_J)$, there exist A',B' such that the path (R',A',B') is a valid execution path and $b'_{y,J+1} = b_{y,J+1}$ for all $y \in D$ (see* (3.25) *for the definition of D).*

Now, assuming the correctness of Claim 3.7.1 and Claim 3.7.2 we present the proof of Lemma 3.6.2.

Proof of Lemma 3.6.2: First, using Claim 3.7.1, we find a zero-distance valid execution path (L'',R'',A'',B). Now, we repeat the procedure described in Claim 3.7.2 on (L'',R'',A'',B), until it becomes a zero-distance monotone execution path, (R',A',B'), which satisfies the conditions of the lemma. □

Now we present the proof of Claim 3.7.1 and Claim 3.7.2.

Proof of Claim 3.7.1: Let $(\ell_j)_{j=1}^J = L$ and $(r_j)_{j=1}^J = R$. Then $\forall j \in [J]$ we define: [15]

$$r'_j := \sqrt{\frac{\ell_j^2 + r_j^2}{2}} \tag{3.47}$$

$$\ell'_j := \sqrt{\frac{\ell_j^2 + r_j^2}{2}} \tag{3.48}$$

and we let $R' := (\ell'_j)_{j=1}^J = (r'_j)_{j=1}^J$. The same as Definition 3.6.3 for all $j \in [J]$, we define

$$c'_j := \sqrt{(\ell'_j)^2 + (r'_j)^2} = \sqrt{2} \cdot r'_j$$

which translates to $c'_j = c_j$. First, we need to show that

$$[0, \ell_j + r_j] \subseteq [0, \ell'_j + r'_j].$$

Note that

$$(\ell'_j + r'_j)^2 - (\ell_j + r_j)^2 = 2c_j^2 - c_j^2 - 2\ell_j r_j \geq 0$$

where the last inequality is due to $c_j = \sqrt{r_j^2 + \ell_j^2}$. One can see that since we can set $a'_{y,0} = a_{y,0}$ for all $y \in D$, it suffices to show that for all $j \in [J]$, $x \in [|\ell_j - r_j| - \delta, \ell_j + r_j]$ (see line 20 of Algorithm 10 and Definition 3.6.3) and $y \in [\ell_j - r_j, \ell_j + r_j]$ such that $y - \Delta/2 \geq x + \Delta$:

$$\frac{4\left(\frac{r_j}{\text{PROJ}(x+\Delta, \ell_j, r_j)}\right)^2 - 1}{4\left(\frac{r_j}{\text{PROJ}(y-\Delta/2, \ell_j, r_j)}\right)^2 - 1} \geq \frac{4(r'_j/(x+\Delta))^2 - 1}{4(r'_j/(y-\Delta/2))^2 - 1} \tag{3.49}$$

We drop the indices j for ease of notation, and let $\alpha := x + \Delta$ and $\beta := y - \Delta/2$. Note that using the formula for PROJECT (see Lemma 3.3.3) have

$$\begin{aligned}
&\frac{4\left(\frac{r}{\text{PROJECT}(\alpha, \ell, r)}\right)^2 - 1}{4\left(\frac{r}{\text{PROJECT}(\beta, \ell, r)}\right)^2 - 1} \cdot \frac{4(r'/\beta)^2 - 1}{4(r'/\alpha)^2 - 1} \\
&= \frac{4\left(\frac{r^2}{\frac{r}{\ell}(\alpha^2 - (\ell - r)^2)}\right) - 1}{4\left(\frac{r^2}{\frac{r}{\ell}(\beta^2 - (\ell - r)^2)}\right) - 1} \cdot \frac{\frac{4r'^2 - \beta^2}{\beta^2}}{\frac{4r'^2 - \alpha^2}{\alpha^2}} \\
&= \frac{(\ell + r)^2 - \alpha^2}{\alpha^2 - (\ell - r)^2} \cdot \frac{\beta^2 - (\ell - r)^2}{(\ell + r)^2 - \beta^2} \cdot \frac{\alpha^2}{\beta^2} \cdot \frac{4r'^2 - \beta^2}{4r'^2 - \alpha^2}
\end{aligned}$$

[15] We define ℓ'_j's for the convenience of the reader, otherwise it is clear that $\ell'_j = r'_j$.

where in the second transition above we used the fact that

$$4\frac{r^2}{\frac{r}{\ell}\left(\alpha^2-(\ell-r)^2\right)}-1=\frac{4r^2\ell^2-\left(\alpha^2-(\ell-r)^2\right)}{\alpha^2-(\ell-r)^2}=\frac{(\ell+r)^2-\alpha^2}{\alpha^2-(\ell-r)^2}$$

and

$$4\frac{r^2}{\frac{r}{\ell}\left(\beta^2-(\ell-r)^2\right)}-1=\frac{4r\ell-(\beta^2-(\ell-r)^2)}{\beta^2-(\ell-r)^2}=\frac{(\ell+r)^2-\beta^2}{\beta^2-(\ell-r)^2}.$$

Now, by re-ordering the factors, and the fact that $4r'^2=2(\ell^2+r^2)$ by (3.47)

$$\begin{aligned}&\frac{(\ell+r)^2-\alpha^2}{\alpha^2-(\ell-r)^2}\cdot\frac{\beta^2-(\ell-r)^2}{(\ell+r)^2-\beta^2}\cdot\frac{\alpha^2}{\beta^2}\cdot\frac{4r'^2-\beta^2}{4r'^2-\alpha^2}\\&=\left(\frac{2(\ell^2+r^2)-\beta^2}{(r+\ell)^2-\beta^2}\cdot\frac{(r+\ell)^2-\alpha^2}{2(\ell^2+r^2)-\alpha^2}\right)\cdot\left(\frac{\alpha^2}{\alpha^2-(r-\ell)^2}\cdot\frac{\beta^2-(r-\ell)^2}{\beta^2}\right)\end{aligned}$$

We bound the two terms above separately. For the first term we have

$$\frac{2(\ell^2+r^2)-\beta^2}{(r+\ell)^2-\beta^2}\cdot\frac{(r+\ell)^2-\alpha^2}{2(\ell^2+r^2)-\alpha^2}=\frac{2(\ell^2+r^2)-\beta^2}{\left(2(r^2+\ell^2)-\beta^2\right)-(\ell-r)^2}\cdot\frac{\left(2(r^2+\ell^2)-\alpha^2\right)-(\ell-r)^2}{2(\ell^2+r^2)-\alpha^2}\geq 1$$

where the inequality follow since for any $0<d<a\leq b$ one has $\frac{a}{a-d}\frac{b-d}{b}\geq 1$. Set $a=2(r^2+\ell^2)-\beta^2$, $b=2(r^2+\ell^2)-\alpha^2$ and $d=(\ell-r)^2$. Note that, $a\leq b$ since $x+\Delta\leq y-\Delta/2$, and $d<a$ since $y-\Delta/2<\ell_j+r_j$.

Now, we bound the second term

$$\left(\frac{\alpha^2}{\alpha^2-(r-\ell)^2}\cdot\frac{\beta^2-(r-\ell)^2}{\beta^2}\right)\geq 1$$

again by the same argument as above, by setting $d=(r-\ell)^2$, $a=\alpha^2$ and $b=\beta^2$. Again, $a\leq b$ since $x+\Delta\leq y-\Delta/2$, and $d<a$ since $x+\Delta>|r-\ell|$ (since $\Delta>\delta$ by the setting of parameters).

Now, combining these two facts (3.49) holds. □

Remark 3.7.1. *One should note that in some cases, the radius of a sphere may decrease when converting it to a zero distance sphere and it means that the size of the band corresponding to orthogonal bands may decrease and this may cause the sphere not being pseudo-random anymore. However, one should note that in our algorithm the radius of the sphere is always* $\Theta(1)$*, meaning that the radius may change by a constant multiplicative factor, so one can re-scale the size of the orthogonal band in the definition (Definition 3.6.3) to cover the previously covered distances.*

Proof of Claim 3.7.2: First note that for $r'\geq r$, we have

$$\frac{4\left(\frac{r}{x+\Delta}\right)^2-1}{4\left(\frac{r}{y-\Delta/2}\right)^2-1}\geq\frac{4\left(\frac{r'}{x+\Delta}\right)^2-1}{4\left(\frac{r'}{y-\Delta/2}\right)^2-1},\tag{3.50}$$

since $f(c) = \frac{4\left(\frac{r}{x+\Delta}\right)^2-1}{4\left(\frac{r}{y-\Delta/2}\right)^2-1}$ is a decreasing function in r, assuming $y-\Delta/2 > x+\Delta$. For the rest of the proof, let $x' := x+\Delta$ and $y' := y-\Delta/2$.

Defining A' and B'. We now construct the sequence of intermediate densities A' that satisfies the conditions in Definition 3.6.3 by modifying the original sequence A on position j (the position where non-monotonicity occurs in the original sequence). Let $a'_{y,i} := a_{y,i}$ and $b'_{y,i} := b_{y,i}$ for all $y \in D$ and $i \in ([J] \cup \{0\}) \setminus \{j\}$. Also, let $b'_{y,J+1} := b_{y,J+1}$ for all $y \in D$. Now, let

$$\forall\, y \in D:\ a'_{y,j} := b_{y,j+1} \tag{3.51}$$

and also set

$$b'_{y,j} := a'_{y,j} \cdot \exp_\mu\left(\frac{4(r_{j+1}/x')^2-1}{4(r_{j+1}/y')^2-1} \cdot \frac{1}{T}\right) = b_{y,j+1} \cdot \exp_\mu\left(\frac{4(r_{j+1}/x')^2-1}{4(r_{j+1}/y')^2-1} \cdot \frac{1}{T}\right) \tag{3.52}$$

since $r'_j = r_{j+1}$.

We now prove that our choice of A' and B' above satisfies the conditions of Definition 3.6.3, i.e. yields a valid execution path. Initial density condition (condition **(1)**) and the terminal density condition (condition **(4)**) are satisfied since they were satisfied by the original execution path (R, A, B), and we did not modify the path on the first and last coordinates. The LSH condition (condition **(3)**) is also satisfied by (3.52) and tha fact that the original execution path satisfied it. We now verify condition **(2)**. Condition **(2a)** follows since $r_{j+1} > r_j$.

Verifying condition (2b). One has, using the assumption that (L, R, A, B) is a valid execution path,

$$\begin{aligned}
b'_{y,j} &= b_{y,j+1} \cdot \exp_\mu\left(\frac{4(r_{j+1}/x')^2-1}{4(r_{j+1}/y')^2-1} \cdot \frac{1}{T}\right) && \text{(by (3.52))}\\
&\le a_{y,j} \cdot \exp_\mu\left(\frac{4(r_{j+1}/x')^2-1}{4(r_{j+1}/y')^2-1} \cdot \frac{1}{T}\right) && \text{(property \textbf{(2b)} for } (R,A,B))\\
&\le b_{y,j} \exp_\mu\left(-\frac{4(r_j/x')^2-1}{4(r_j/y')^2-1} \cdot \frac{1}{T}\right) \cdot \exp_\mu\left(\frac{4(r_{j+1}/x')^2-1}{4(r_{j+1}/y')^2-1} \cdot \frac{1}{T}\right) && \text{(property \textbf{(3)} for } (R,A,B))\\
&\le b_{y,j} && \text{(by (3.50) together with } r_j < r_{j+1})\\
&\le a_{y,j-1} && \text{(property \textbf{(2b)} for } (R,A,B))
\end{aligned}$$

Verifying condition (2c). We need to prove

$$\sum_{y \in [0, r_{j+1}(\sqrt{2}-\psi)] \cap D} b'_{y,j} \le \frac{\tau}{1-2\tau} \cdot \sum_{y \in (r_{j+1}(\sqrt{2}-\psi), r_{j+1}(\sqrt{2}+\psi)) \cap D} b'_{y,j} \tag{3.53}$$

Note that by property **(2c)** we have

$$\sum_{y\in[0,r_{j+1}(\sqrt{2}-\psi)]\cap D} b_{y,j+1} \leq \frac{\tau}{1-2\tau}\cdot \sum_{y\in(r_{j+1}(\sqrt{2}-\psi),r_{j+1}(\sqrt{2}+\psi))\cap D} b_{y,j+1} \tag{3.54}$$

Also, recall that by (3.52) we have

$$b'_{y,j} = b_{y,j+1}\cdot \exp_\mu\left(\frac{4(r_{j+1}/x')^2-1}{4(r_{j+1}/y')^2-1}\cdot\frac{1}{T}\right)$$

Now, combing the fact that $\exp_\mu\left(\frac{4(r_{j+1}/x')^2-1}{4(r_{j+1}/y')^2-1}\cdot\frac{1}{T}\right)$ is increasing in y' with (3.54), proves (3.53).

We have thus shown that (R', A', B') is a valid execution path. Note that $b'_{y,J+1} = b_{y,J+1}$ for all $y \in D$ by definition of b', as required. □

3.8 Feasible LP solutions based on valid execution paths

First, we state the main result of this section informally below. We refer the reader to Claim 3.8.6 for the formal version of this claim.

Claim 3.8.1. *(Informal) If the length of a valid execution path is large enough, then the terminal densities must be small.*

We prove this claim, by arguing that if the terminal densities are not small then there exists a feasible solution to the LP. However, the feasible solution that we construct, has a cost larger than the optimal solution of the LP, which is a contradiction. This implies that we cannot have large terminal densities.

We use Definition 3.6.6, Definition 3.6.7 and the corresponding notations in the rest of this section. At this point, one should recall that the definition of valid execution paths (Definition 3.6.3) is over the continuous densities. Now, we need to present a similar notion for *discretized log-densities*.

Claim 3.8.2 (Discretized execution path)**.** *If the $f_{z_i,j}$ variables are defined as per* (3.32) *(based on a zero distance monotone execution path (R, A, B), with $J = |R|$) then*

(1) Initial densities: *For any integer i: $f_{z_i,1} \leq \min\left\{\frac{z_i^2-x^2}{2}+3\delta_z, 1-\frac{x^2}{2}\right\}$.*

(2) Truncation: *for any $j \in [J]$ and $i \in \{k_j+1,\ldots,I\}$ one has $f_{z_i,j} \leq f_{z_{k_j},j} + \log_{1/\mu}\frac{2-2\tau}{1-2\tau}$.*

(3) Locality Sensitive Hashing: *for any $j \in [J]$ and any integer $i \in \{k_j,\ldots,I\}$ one has $f_{z_i,j+1} \leq f_{z_i,j} - \frac{2(z_{k_j}/x)^2-1}{2(z_{k_j}/z_i)^2-1}\cdot\frac{1}{T}\cdot(1-10^{-4})$.*

Proof. For the purposes of the proof it is convenient to introduce an auxiliary definition. For every $j \in [J]$ and every integer $i \in \{k_j,\ldots,I\}$ define

$$\widetilde{a}_{z_i,j} := \sum_{y \in D_x \cap [z_{i+1}, z_{i-1})} a_{y,j}. \tag{3.55}$$

and

$$\widetilde{b}_{z_i,j} := \sum_{y \in D_x \cap [z_{i+1}, z_{i-1})} b_{y,j}. \tag{3.56}$$

Note that with these definitions in place (3.32) is equivalent to

$$f_{z_i,j} := \log_{1/\mu} \widetilde{b}_{z_i,j}. \tag{3.57}$$

We also let $D := D_x$, omitting the dependence on x, to simplify notation. We now prove the properties one by one.

(1) Initial densities condition: First, note that by the initial densities condition for the execution path (R, A, B) together with the truncation conditions (Definition 3.6.3) one has

$$\sum_{y' \in [0,y] \cap D} b_{y',1} \leq \min\left\{\exp_\mu\left(\frac{y^2 - x^2}{2}\right), \exp_\mu\left(\frac{1-x^2}{2}\right)\right\}.$$

Combining this with (3.56), we get

$$\begin{aligned}
\widetilde{b}_{z_i,1} &= \sum_{y \in D \cap (z_{i+1}, z_{i-1})} b_{y,1} \leq \sum_{y \in [0, z_{i-1}] \cap D} b_{y,1} \\
&\leq \min\left\{\exp_\mu\left(\frac{z_{i-1}^2 - x^2}{2}\right), \exp_\mu\left(\frac{1-x^2}{2}\right)\right\} \\
&= \min\left\{\exp_\mu\left(\frac{(1+\delta_z)^2 \cdot z_i^2 - x^2}{2}\right), \exp_\mu\left(1 - \frac{x^2}{2}\right)\right\} \\
&\leq \min\left\{\exp_\mu\left(\frac{\cdot z_i^2 - x^2}{2} + 3\delta_z\right), \exp_\mu\left(1 - \frac{x^2}{2}\right)\right\},
\end{aligned}$$

where we used the definition of the grid Z, the fact that $\mu = o(1)$ and that for $z_i \geq \sqrt{2}$ the second term is the minimum term.

(2) Truncation conditions (effect of PSEUDORANDOMIFY): We have, using (3.56),

$$\begin{aligned}\sum_{i=k_j+1}^{I} \widetilde{b}_{z_i,j} &\le 2 \sum_{y\in D\cap(0,z_{k_j+1})} b_{y,j} + \sum_{y\in D\cap(z_{k_j+1},z_{k_j})} b_{y,j} \\ &\le 2 \sum_{y\in D\cap(0,z_{k_j})} b_{y,j} \\ &\le 2 \sum_{y\in D\cap(0,r_j(\sqrt{2}+\psi))} b_{y,j}.\end{aligned} \tag{3.58}$$

The last transition uses the fact that by definition of k_j (see (3.31)) we have $r_j\cdot(\sqrt{2}+\psi)\in[z_{k_j},z_{k_j-1})$, and in particular, $r_j\cdot(\sqrt{2}+\psi)\ge z_{k_j}$.

We now note that since $10\psi\le\delta_z=10^{-6}$ by assumption of the claim and $z_{k_j+1}=(1+\delta_z)^{-1}z_{k_j}$, we further have

$$(r_j(\sqrt{2}-\psi),r_j(\sqrt{2}+\psi))\subset[z_{k_j+1},z_{k_j-1})$$

which implies

$$\sum_{y\in D\cap(r_j(\sqrt{2}-\psi),r_j(\sqrt{2}+\psi))} b_{y,j}\le\widetilde{b}_{z_{k_j},j}. \tag{3.59}$$

At the same time, since (R,A,B) was a valid execution path, then by property **(2c)** in Definition 3.6.3, we have

$$\begin{aligned}\sum_{y\in D\cap(0,r_j(\sqrt{2}+\psi))} b_{y,j} &\le \left(1+\frac{\tau}{1-2\tau}\right) \sum_{y\in D\cap(r_j(\sqrt{2}-\psi),r_j(\sqrt{2}+\psi))} b_{y,j} \\ &= \frac{1-\tau}{1-2\tau} \sum_{y\in D\cap(r_j(\sqrt{2}-\psi),r_j(\sqrt{2}+\psi))} b_{y,j}.\end{aligned}$$

Substituting the bound above into (3.58) and using (3.59) yields

$$\sum_{i=k_j+1}^{I}\widetilde{b}_{z_i,j}\le\frac{2-2\tau}{1-2\tau}\cdot\widetilde{b}_{z_{k_j},j},$$

establishing the claim.

(3) LSH conditions: For all $j\in[J]$ let $c_j:=\sqrt{2}r_j$. One can think of c_j as the distance from a query on the surface of the j-th sphere in the execution path to a 'typical' point on the sphere. Note that (3.31) defines a rounding of c_j's points on the grid D. Specifically, c_j is rounded to z_{k_j}'s.

Claim 3.8.3. *Let $x':=x+\Delta$ and let $y\in(z_{i+1},z_{i-1}]$, if $y'=y-\Delta/2$, and $i\in\{k_j,\dots,I\}$ then we have the*

following claim.

$$-\frac{4\left(\frac{r_j}{x'}\right)^2-1}{4\left(\frac{r_j}{y'}\right)^2-1} \le -\frac{2\left(\frac{z_{k_j}}{x}\right)^2-1}{2\left(\frac{z_{k_j}}{z_i}\right)^2-1}(1-10^{-4})$$

We prove this claim in Appendix C.6.

By property **(3)** in Definition 3.6.3, one has

$$a_{y,j} \le b_{y,j} \cdot \exp_\mu\left(-\frac{2(c_j/x')^2-1}{2(c_j/y')^2-1} \cdot \frac{1}{T}\right). \tag{3.60}$$

Thus, for all $i \in \{k_j, \dots, I\}$ we have

$$\begin{aligned}
\widetilde{b}_{z_i,j+1} &= \sum_{y \in D \cap (z_{i+1}, z_{i-1})} b_{y,j+1} && \text{By (3.56)} \\
&\le \sum_{y \in D \cap (z_{i+1}, z_{i-1})} a_{y,j} && \text{Property \textbf{(2b)} for } (R, A, B) \\
&\le \sum_{y \in D \cap (z_{i+1}, z_{i-1})} b_{y,j} \cdot \exp_\mu\left(-\frac{2(c_j/x')^2-1}{2(c_j/y')^2-1} \cdot \frac{1}{T}\right) && \text{By (3.60)} \\
&\le \widetilde{b}_{z_i,j} \cdot \exp_\mu\left(-\frac{2(z_{k_j}/x)^2-1}{2(z_{k_j}/z_i)^2-1} \cdot \frac{1}{T} \cdot (1-10^{-4})\right) && \text{By (3.56) and Claim 3.8.3}
\end{aligned}$$

This completes the proof of **(3)**. □

3.8.1 Construction of a feasible solution

In this section, we construct a feasible solution to the LP, i.e., $g_{y,j}$'s and α_j's, based on the execution path that we are considering. Later, we show the relation between the cost of this solution and the length of the execution path.

First, letting $J = |R|$, recall that $R = (r_j)_{j=1}^J$. Then, for all $s \in [J]$ define $c_s := \sqrt{2} \cdot r_s$ and let $c_0 = +\infty$ for convenience. Let $\widetilde{T}$ be such that

$$\frac{1}{T} \cdot (1-10^{-4}) = \frac{1}{\widetilde{T}}. \tag{3.61}$$

Let $x' = x + \Delta$. We classify steps $s = 1, \dots, J$ into three types:

- We say that a step s is **stationary** if $c_s = c_{s-1}$ (this corresponds to the algorithm performing multiple rounds of hashing on the same sphere).
- Otherwise we call step s **minor**, if $\frac{c_s}{c_{s-1}} \ge 1 - \frac{1}{\sqrt{T}}$,
- and call step s **major**, otherwise.

Let $R = R_{stat} \cup R_m \cup R_M$ denote the partition of R into stationary, minor and major steps. Let $j_1, \ldots, j_{|R_m \cup R_M|}$ be such that $z_{j_1} > z_{j_2} > \ldots > z_{j_{|R_m \cup R_M|}}$ are exactly the c_s values corresponding to non-stationary steps, in decreasing order.

Note that by Lemma 3.5.1 and parameter settings in the algorithm, $c_1 \leq R_{max}\sqrt{2} = O(1)$. Since the grid D (see (3.25)) contains only elements at least as large as $x + 1.5\Delta$, and if we let x to be lower bounded by an absolute constant we have $|R_M| = O(\sqrt{T})$. The reason is that by the definition above, for any major step s, we have

$$\frac{c_s}{c_{s-1}} < 1 - \frac{1}{\sqrt{T}}.$$

We define the feasible solution $g_{y,j}$'s and α_j's to LP(x, j^*) as defined in (3.33) without the **non-empty range constraint**. We construct feasible $g_{y,j}$ and α_j by induction on $j = 1, \ldots, j^*$. It will be important that the constructed solutions for $g_{y,j}$'s are non-decreasing in y for $y \in [0, z_j]$ for every $j = 1, \ldots, j^*$.

For the rest of the section, whenever we are working with discrete functions and it is clear from the context, we drop the condition $y \in Z_x$.

On the other hand, it is more convenient to work with the following formulation of the LP constraints, since we construct the solution in an inductive way.

$$\forall y : g_{y,1} \leq \min\left\{\frac{y^2 - x^2}{2}, 1 - \frac{x^2}{2}\right\}$$

for all $j < j^*, y < z_j$:

$$g_{y,j+1} \leq \min\left\{g_{y,j} - \frac{2\left(z_j/x\right)^2 - 1}{2\left(z_j/y\right)^2 - 1} \cdot \alpha_j,\ g_{z_{j+1},j} - \frac{2\left(z_j/x\right)^2 - 1}{2(z_j/z_{j+1})^2 - 1} \cdot \alpha_j\right\}$$

Base: For all $j \in [j_1]$ such that $y \leq z_j$, we set

$$g_{y,j} := \min\left\{\frac{y^2 - x^2}{2}, 1 - \frac{x^2}{2}\right\}. \tag{3.62}$$

We let $\alpha_j := 0$ for all $j \in [j_1 - 1]$ so that **spherical LSH** constraints of the LP in (3.33) are satisfied for $j \in [j_1 - 1]$. That is, we don't have any progress using spherical LSH, since $\alpha_j = 0$ for all $j \in [j_1 - 1]$. The **truncation** constraints of the LP in (3.33) are satisfied since the rhs of (3.62) is non-decreasing in y.

Inductive step $j_i \to j_i + 1, \ldots, j_{i+1}$: We let $a := j_i$ and $b := j_{i+1}$ to simplify notation. Let s be the first step on sphere z_a, i.e., $c_s = z_a$ and $c_{s-1} \neq z_a$. Also, let N be the number of steps that we stay on sphere z_a, i.e.,

$$c_s = c_{s+1} = \ldots = c_{s+N-1} = z_a \text{ and } c_{s+N} \neq z_a.$$

Note that steps $s + 1, \ldots, s + N - 1$ are stationary as per our definitions.

It is convenient to define a sequence of auxiliary variables in order to handle the sequence of $N-1$ stationary steps (note that $N-1$ could be zero).

Upper bounds $h_y^{(q)}$, $q = 0,\ldots,N$, on density after (possible) stationary steps:

For all $y \le z_a$: (Starting density)

$$h_y^{(0)} := g_{y,a} \tag{3.63}$$

For all $q \in [N-1]$, $y \le z_a$: (Stationary steps)

$$h_y^{(q)} := \min\left\{ h_y^{(q-1)} - \frac{2(z_a/x)^2-1}{2(z_a/y)^2-1}\frac{1}{\widetilde{T}},\ h_{z_a}^{(q-1)} - \frac{2(z_a/x)^2-1}{\widetilde{T}} \right\} \tag{3.64}$$

For $y \le z_a$: (Final density)

$$h_y^{(N)} := h_y^{(N-1)} - \frac{2(z_a/x)^2-1}{2(z_a/y)^2-1}\frac{1}{\widetilde{T}}. \tag{3.65}$$

Equipped with the definitions of h above, we now define g to satisfy the inductive step. First let $\alpha_a := \frac{N}{\widetilde{T}}$ and let $\alpha_{a+s} := 0$ for $s = 1,\ldots,N-1$. Then define for all $y \le z_b$:

$$g_{y,a+1} := \min\left\{ h_y^{(N)},\ h_{z_{a+1}}^{(N)} \right\}$$

and for $j = a+2,\ldots,b$ and all $y \le z_j$ let

$$g_{y,j} := \min\{g_{y,j-1}, g_{z_j,j-1}\}.$$

We note that this implies for all $y \le z_b$

$$g_{y,b} := \min\left\{ \min_{j\in\{a+1,\ldots,b\}} \left\{ h_{z_j}^{(N)} \right\}, h_y^{(N)} \right\}. \tag{3.66}$$

This finished the inductive step.

Note that in the last step, i.e., when $z_a = z_{j^*-1}$, since we do not have truncation condition for g_{y,j^*}'s, we define

$$\forall\, y \le z_{j^*-1} :\ g_{y,j^*} = h_y^{(N)}. \tag{3.67}$$

3.8.2 Monotonicity claims

Claim 3.8.4 (Unique maximum after LSH)**.** *For every integer $t \ge 1$, $x \in (0,\sqrt{2})$ and any sequence $c_1 \ge c_2 \ge \ldots \ge c_t \ge x$, such that*

$$f(y) = \frac{y^2-x^2}{2} - \sum_{s=1}^{t} \frac{2(c_s/x)^2-1}{2(c_s/y)^2-1}\cdot\frac{1}{T}$$

satisfies $f(\sqrt{2}c_t) > 0$*, the following conditions hold. There exists* $y^* \in (x, \sqrt{2}c_t]$ *such that the function satisfies* $f(y^*) = 0$ *is monotone increasing on the interval* $[y^*, \eta]$*, where* η *is where the (unique) maximum of* f *on* $(y^*, \sqrt{2}c_t]$ *happens.*

Proof. We prove that $\frac{\partial^2 f(y)}{\partial y^2}$ is a monotone decreasing function. One should note that

$$\frac{\partial^2 f(y)}{\partial y^2} = 1 - \sum_{s=1}^{t} \frac{4c_s^2(2c_s^2+3y^2)}{(2c_s^2-y^2)^3} \cdot \frac{2(c_s/x)^2-1}{T}$$

Now, one can see that $\frac{\partial^2 f(y)}{\partial y^2}$ is a monotone decreasing function in y. We then note that $f(x) \le 0$, and the function $f(y)$ has exactly one maximum on $(y^*, \sqrt{2}c_s)$. □

We will need

Claim 3.8.5 (Monotonicity)**.** *For every* $i \in [|R|]$ *we have*

(a) *If* $g_{z_{j_i}, j_i} > 0$ *then there exists a* $y^* \in (x, \sqrt{2})$ *such that* $g_{y^*, j_i} \ge 0$*,* $g_{y, j_i} \le 0$ *for any* $y \in Z_x$ *such that* $y \le y^*$*, and* g_{y, j_i} *is non-decreasing in* y *for* $y \in [y^*, z_{j_i}]$*;*

(b) *If* $h^{(N-1)}_{z_{j_i}} > 0$ *then there exists a* $y^* \in (x, \sqrt{2})$ *such that* $h^{(N-1)}_{y^*} \ge 0$*,* $h^{(N-1)}_{y} \le 0$ *for any* $y \in Z_x$ *such that* $y \le y^*$ *and* $h^{(N-1)}_{y}$ *is non-decreasing in* y *for* $y \in [y^*, z_{j_i}]$*.*

Proof. Let

$$q(y) := \sum_{i=1}^{t} \frac{2(c_s/x)^2-1}{2(c_s/y)^2-1} \frac{1}{T},$$

where $c_1 \ge c_2 \ge \ldots \ge c_t \ge z_1 \ge x$ for some $z_1 \ge x$. And let y_1^* be such that $\frac{(y_1^*)^2-x^2}{2} - q(y_1^*) = 0$ and let $\widetilde{y}_1$ be the smallest value such that $\widetilde{y}_1 \ge y_1^*$ and $\frac{\widetilde{y}_1^2-x^2}{2} - q(\widetilde{y}_1) = \theta$ for some $\theta \ge 0$. Now define $G_1(y)$ on $[y_1^*, z_1]$, for some $z_1 \ge \widetilde{y}_1$ as follows

$$G_1(y) := \begin{cases} \frac{y^2-x^2}{2} - q(y) & y \in [y_1^*, \widetilde{y}_1) \\ \theta & y \in [\widetilde{y}_1, z_1] \end{cases} \tag{3.68}$$

See the red curve in Figure 3.11.

Also, let $\hat{q}(y) := \frac{2(z_1/x)^2-1}{2(z_1/y)^2-1} \frac{1}{T}$. Let $y_2^* \ge y_1^*$ such that $G_1(y_2^*) - \hat{q}(y_2^*) = 0$. Now, we define $G_2(y)$ for $y \in [y_2^*, z_2]$ as follows:

$$G_2(y) := \min\left\{G_1(y) - \hat{q}(y), \theta'\right\}$$

where $\theta' := G_1(z_2) - \hat{q}(z_2)$ and $\theta' \ge 0$ for some $z_2 \le z_1$. By the definition of y_2^*, function $G_2(y)$ for $y \in [y_2^*, \widetilde{y}_1]$ is in the form of the function in Claim 3.8.4 and thus, it has a unique maximum at some $\eta \in [y_2^*, \widetilde{y}_1]$. Also,

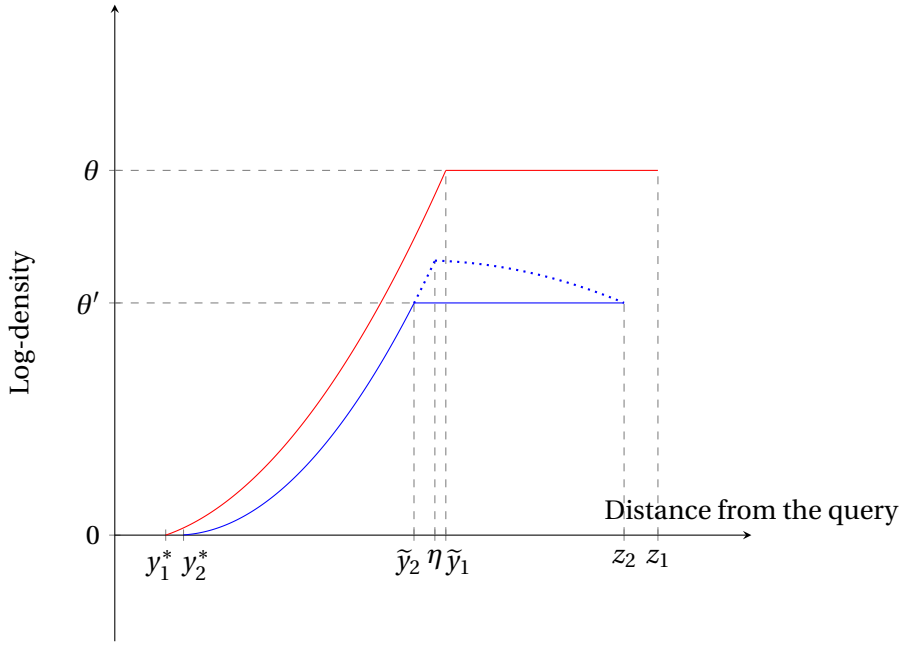


Figure 3.11: An illustration of proof of Claim 3.8.5. The red and blue curves represent functions G_1 and G_2. The dotted part of the blue curve represents $G_1 - \hat{q}$ function for interval $[\widetilde{y}_2, z_2]$, which gets truncated by θ'.

recall that $G_1(y) = \theta$ for $y \in [\widetilde{y}_1, z_2]$. Also, one should note that since $\hat{q}(y)$ is a monotone increasing function for $y \in (0, \sqrt{2}z_1)$ and hence for $y \in [\widetilde{y}_1, z_2]$, then $\theta' \leq G_2(\widetilde{y}_1)$ and therefore $\theta' \leq G_2(\eta)$. This guarantees that there exist a $\widetilde{y}_2 \in [y_2^*, \eta]$ such that $G_2(\widetilde{y}_2) = \theta'$. The reason is that $G_2(y)$ is a continuous increasing function for $y \in [\widetilde{y}_2, \eta]$. So, we have

$$G_2(y) := \begin{cases} \frac{y^2 - x^2}{2} - q'(y) & y \in [y_2^*, \widetilde{y}_2) \\ \theta' & y \in [\widetilde{y}_2, z_2] \end{cases} \tag{3.69}$$

where, $q'(y) := q(y) - \hat{q}(y)$. See the blue curve in Figure 3.11. Now, one can see that by a simple inductive argument starting with the initial densities

$$\min\left\{\frac{y^2 - x^2}{2}, 1 - \frac{x^2}{2}\right\}$$

which is in the form of (3.68), the statement of the claim holds. □

3.8.3 Bounding terminal densities using feasible LP solutions

Claim 3.8.6 (Feasible LP solution from an execution path)**.** *If integer J is such that $J > \frac{T}{1-10^{-4}}\mathrm{OPT(LP)}$ then, for all $y \leq z_{j^*-1}$, $f_{y,J+1} < 7\delta_z$ for $j^* = k_J + 1$ (see Definition 3.6.7 for the definition of k_J).*

Proof. We prove the claim by contradiction. Suppose that there exists $z \leq z_{j^*-1}$ such that $f_{z,J+1} \geq 7\delta_z$.

We define the feasible solution $g_{y,j}$'s and α_j's to LP(x, j^*) as defined in (3.33). However, the cost of this solution will be more than the optimal cost of the LP, which gives us the contradiction. First, we construct the solution without considering the **non-empty range constraint**. Then, we show that applying $f_{z,J+1} \geq 7\delta_z$ the **non-empty range constraint** is satisfied too.

Let $g_{y,j}$ and α_j be defined as by induction on $j = 1, \ldots, j^*$ as above. We prove by induction on j that if there exists a s such that $c_s = z_j$ and $c_{s-1} \neq z_j$ then for all $y \leq z_j$,

$$f_{y,s} \leq g_{y,j} + X_s, \tag{3.70}$$

where we define

$$X_s := \sum_{r \in R_m \text{ s.t. } r \leq s} O\left(\frac{1}{\sqrt{T}}\right) \cdot \frac{1}{\widetilde{T}} + \sum_{r \in R_M \text{ s.t. } r \leq s} \frac{O(1)}{\widetilde{T}} + s \cdot \delta + 3\delta_z. \tag{3.71}$$

for ease of notation, and let $\delta = \log_{1/\mu} \frac{2-2\tau}{1-2\tau} = \Theta\left(\frac{1}{\log 1/\mu}\right)$. One should note that the δ used in this proof is not related to the δ used in the algorithm. Also, let $c_0 = \infty$ and $c_{J+1} = z_{j^*}$, for easing the corner case analysis.

Base: For all $j \in \{1, 2, \ldots, j_1\} = [j_1]$ and all $y \leq z_j$, in (3.62) we did set

$$g_{y,j} := \min\left\{\frac{y^2 - x^2}{2}, 1 - \frac{x^2}{2}\right\}. \tag{3.72}$$

Also, recall that we let $\alpha_j := 0$ for all $j \in [j_1 - 1]$ (see **base case** in Section 3.8.1). Now, note that we have $f_{y,1} \leq g_{y,1} + 3\delta_z$ for all y by Claim 3.8.2, **(1)** combined with the assumption that $\delta_z \leq 1$. So, the base holds.

Inductive step $j_i \to j_i + 1, \ldots, j_{i+1}$**:** We let $a := j_i$ and $b := j_{i+1}$ to simplify notation. Let s be the first step on sphere z_a: $c_s = z_a$ and $c_{s-1} \neq z_a$. Also, let N be the number of steps that we stay on sphere z_a, i.e.,

$$c_s = c_{s+1} = \ldots = c_{s+N-1} = z_a \text{ and } c_{s+N} \neq z_a.$$

Note that steps $s+1, \ldots, s+N-1$ are stationary as per our definitions. We let $t := s + N$ for convenience. By the inductive hypothesis for any $y \leq z_a$ we have

$$f_{y,s} \leq g_{y,a} + X_s \tag{3.73}$$

We prove that for any $y \leq z_b$

$$f_{y,t} \leq g_{y,b} + X_t. \tag{3.74}$$

Let $h_y^{(q)}, q = 0, \ldots, N$ and $g_{y,j}, j = a, \ldots, b$, be defined as above. We now upper bound $f_{y,s+q}$ in terms of $f_{y,s+(q-1)}$. We have for all $q \in [N-1]$ and $y \leq z_b$:

$$\begin{aligned} f_{y,s+q} &\leq \min\left\{f_{y,s+(q-1)} - \frac{2(z_a/x)^2 - 1}{2(z_a/y)^2 - 1}\frac{1}{\widetilde{T}},\ f_{z_a,s+(q-1)} - \frac{2(z_a/x)^2 - 1}{\widetilde{T}} + \delta\right\} \\ &\leq \min\left\{f_{y,s+(q-1)} - \frac{2(z_a/x)^2 - 1}{2(z_a/y)^2 - 1}\frac{1}{\widetilde{T}},\ f_{z_a,s+(q-1)} - \frac{2(z_a/x)^2 - 1}{\widetilde{T}}\right\} + \delta. \end{aligned} \tag{3.75}$$

where the first transition is by Claim 3.8.2. Similarly we have (again by Claim 3.8.2)

$$\begin{aligned} f_{y,t} &\le \min\left\{ f_{y,s+(N-1)} - \frac{2(z_a/x)^2-1}{2(z_a/y)^2-1}\frac{1}{\widetilde{T}},\ f_{z_{j_b},s+(N-1)} - \frac{2(z_a/x)^2-1}{2(z_a/z_b)^2-1}\frac{1}{\widetilde{T}} + \delta \right\} \\ &\le \min\left\{ f_{y,s+(N-1)} - \frac{2(z_a/x)^2-1}{2(z_a/y)^2-1}\frac{1}{\widetilde{T}},\ f_{z_{j_b},s+(N-1)} - \frac{2(z_a/x)^2-1}{2(z_a/z_b)^2-1}\frac{1}{\widetilde{T}} \right\} + \delta \end{aligned} \tag{3.76}$$

We now note that the recurrence relations (3.64) and (3.65) defining $h_y^{(q)}$ are only different from the above by an additive δ term, and the initial condition (3.63) for $h_y^{(0)}$ is only different from the inductive hypothesis (3.73) by an additive X_s term. Combining these observations, we get

$$f_{y,t} \le \min\left\{ h_y^{(N)}, h_{z_b}^{(N)} \right\} + X_s + \delta\cdot(t-s). \tag{3.77}$$

Now, one can see that we have the following upper bound for X_s for any s using the definition of X_s

$$\begin{aligned} X_s &= \sum_{r\in R_m \text{ s.t. } r\le s} O\left(\frac{1}{\sqrt{T}}\right)\cdot\frac{1}{\widetilde{T}} + \sum_{r\in R_M \text{ s.t. } r\le s} \frac{O(1)}{\widetilde{T}} + s\cdot\delta + 3\delta_z \\ &\le O\left(\frac{1}{\sqrt{T}}\right) + 3\delta_z \end{aligned} \tag{3.78}$$

since we have at most $O(\sqrt{T})$ major steps, at most $O(T)$ minor steps, and $\delta = \Theta\left(\frac{1}{T^2}\right)$. This implies

$$f_{y,t} \le \min\left\{ h_y^{(N)}, h_{z_b}^{(N)} \right\} + 3\delta_z + O\left(\frac{1}{\sqrt{T}}\right) \le h_y^{(N)} + 4\delta_z. \tag{3.79}$$

Combining this with the assumption that there exists a $z \le z_{j^*-1}$ such that $f_{z,J+1} \ge 7\delta_z$ we have

$$h_y^{(N)} \ge 0 \text{ for all } y \ge z_{j^*-1}, \tag{3.80}$$

which we prove below and will be useful whenever we want to invoke Claim 3.8.5.

Claim 3.8.7. *$\forall\, y \ge z_{j^*-1}$, we have $h_y^{(N)} \ge 0$.*

Proof. $\forall\, y \ge z_{j^*}:\ h_y^{(N)} \ge 0$. Assume that there exists a $z \le z_{j^*-1}$ such that $f_{z,J+1} \ge 7\delta_z$. Now, by the fact that $f_{y,j}$'s are monotone in j, and by (3.79) we have

$$7\delta_z \le f_{z,J+1} \le f_{z,t} \le h_z^{(N)} + 4\delta_z,$$

which implies

$$3\delta_z \le h_z^{(N)}.$$

On the other hand, by (3.64) we get

$$h_z^{(N-1)} \le h_{z_a}^{(N-1)}.$$

which implies $h_{z_a}^{(N-1)} \ge 3\delta_z \ge 0$. Thus, by Claim 3.8.5, $h_y^{(N-1)}$ is non-decreasing in $[z, z_a]$. So, for any

$y \in [z, z_a]$,

$$3\delta_z \leq h_y^{(N-1)} \tag{3.81}$$

Also, by (3.65) we have

$$h_y^{(N)} = h_y^{(N-1)} - \frac{2(z_a/x)^2 - 1}{2(z_a/y)^2 - 1} \frac{1}{\widetilde{T}} = h_y^{(N-1)} - O\left(\frac{1}{T}\right) \tag{3.82}$$

Combing (3.81) and (3.82), we prove that for $y \geq z_{j^*-1}$:

$$h_y^{(N)} \geq 2\delta_z \geq 0.$$

□

The following characterization of $X_t - X_s$ will be useful:

$$X_t - X_s = \delta \cdot (t-s) + \begin{cases} O\left(\frac{1}{\sqrt{T}}\right) \cdot \frac{1}{\widetilde{T}} & \text{if } t \in R_m \\ \frac{O(1)}{\widetilde{T}} & \text{if } t \in R_M. \end{cases} \tag{3.83}$$

The above follows by (3.71) since all steps between s and t are stationary.

We now the upper bound the minimum on the rhs in (3.77). Recall from (3.66) that

$$g_{y,b} := \min\left\{ \min_{j \in \{a+1,\ldots,b\}} \left\{ h_{z_j}^{(N)} \right\}, h_y^{(N)} \right\}.$$

Let y'' be such that $g_{y,b} = h_{y''}^{(N)}$. We consider two cases, depending on whether $y'' = y$.

Case 1: $y = y''$ **(the simple case).** In that case we have

$$\begin{aligned} f_{y,t} &\leq \min\left\{ h_y^{(N)}, h_b^{(N)} \right\} + X_s + \delta \cdot (t-s) && \text{By (3.77)} \\ &= h_y^{(N)} + X_s + \delta \cdot (t-s) && \text{Since } y'' = y \\ &= g_{y,b} + X_s + \delta \cdot (t-s) && \text{Combining } y'' = y \text{ and (3.66)} \\ &\leq g_{y,b} + X_t, && \text{By (3.83)} \end{aligned}$$

as required.

Case 2: $y \neq y''$ **(the main case).**

$$\begin{aligned} f_{y,t} &\leq \min\left\{ h_y^{(N)}, h_{z_b}^{(N)} \right\} + X_s + \delta \cdot (t-s) \\ &\leq h_{z_b}^{(N)} + X_s + \delta \cdot (t-s) \\ &= g_{y,b} + X_s + \delta \cdot (t-s) + (h_{z_b}^{(N)} - g_{y,b}). \end{aligned} \tag{3.84}$$

In what follows we show that

$$h^{(N)}_{z_b} - g_{y,b} = \begin{cases} O\left(\frac{1}{\sqrt{T}}\right) \cdot \frac{1}{\widetilde{T}} & \text{if } t \in R_m \\ \frac{O(1)}{\widetilde{T}} & \text{if } t \in R_M, \end{cases} \tag{3.85}$$

which gives the result once substituted in (3.84), as per (3.83).

We now consider two case, depending on whether t is a *minor* or a *major* step. For both steps we use the fact that $y'' \neq y$ implies $y'' \geq z_b$ (this follows by definition of y'' together with (3.66)).

Minor steps ($t \in R_m$). In this case we have

$$\begin{aligned}
& h^{(N)}_{z_b} - g_{y,b} \\
=& h^{(N)}_{z_b} - h^{(N)}_{y''} && \text{By definition of } y'' \\
=& h^{(N-1)}_{z_b} - \frac{2(z_a/x)^2-1}{2(z_a/z_b)^2-1} \cdot \frac{1}{\widetilde{T}} - h^{(N-1)}_{y''} + \frac{2(z_a/x)^2-1}{2(z_a/y'')^2-1} \cdot \frac{1}{\widetilde{T}} && \text{By (3.65)} \\
\leq& \frac{2(z_a/x)^2-1}{2(z_a/y'')^2-1} \cdot \frac{1}{\widetilde{T}} - \frac{2(z_a/x)^2-1}{2(z_a/z_b)^2-1} \cdot \frac{1}{\widetilde{T}}.
\end{aligned}$$

The last transition used Claim 3.8.5, **(b)**, and the fact that $y'' \geq z_b$: we only need to verify the preconditions of Claim 3.8.5, which follows by (3.80) together with the fact that $h^{(N-1)}_y \geq h^{(N)}_y$ for all y.

We now bound the rhs of the equation above by

$$\begin{aligned}
& \frac{2(z_a/x)^2-1}{2(z_a/y'')^2-1} \cdot \frac{1}{\widetilde{T}} - \frac{2(z_a/x)^2-1}{2(z_a/z_b)^2-1} \cdot \frac{1}{\widetilde{T}} \\
\leq& \frac{2(z_a/x)^2-1}{\widetilde{T}} \left(1 - \frac{1}{2(z_a/z_b)^2-1}\right) && \text{Since } y'' \leq z_a \\
=& \frac{2(z_a/x)^2-1}{\widetilde{T}} \left(1 - \frac{1}{2(1-1/\sqrt{T})^{-2}-1}\right) && \text{Since this is a } \textit{minor} \text{ step} \\
=& \frac{2(z_a/x)^2-1}{\widetilde{T}} \cdot O(1/\sqrt{T}) \\
\leq& \frac{2(z_a/\Delta)^2-1}{\widetilde{T}} \cdot O(1/\sqrt{T}) \\
=& \frac{1}{\widetilde{T}} \cdot O\left(\frac{1}{\sqrt{T}}\right), && \text{Since } z_a \leq R_{max} = O(1) \text{ and } \Delta = \Omega(1)
\end{aligned}$$

Major steps ($t \in R_M$). Now, we consider the case when the step is *major*.

$$\begin{aligned}
& h^{(N)}_{z_b} - g_{z_b,b} \\
=& h^{(N)}_{z_b} - h^{(N)}_{y''} && \text{By definition of } y'' \text{ and (3.66)} \\
=& h^{(N-1)}_{z_b} - \frac{2(z_a/x)^2-1}{2(z_a/z_b)^2-1}\cdot\frac{1}{\widetilde{T}} - h^{(N-1)}_{y''} + \frac{2(z_a/x)^2-1}{2(z_a/y'')^2-1}\cdot\frac{1}{\widetilde{T}} && \text{By (3.65)} \\
\leq& \frac{2(z_a/x)^2-1}{2(z_a/y'')^2-1}\cdot\frac{1}{\widetilde{T}} - \frac{2(z_a/x)^2-1}{2(z_a/z_b)^2-1}\cdot\frac{1}{\widetilde{T}}.
\end{aligned}$$

The last transition used Claim 3.8.5, **(b)**, and the fact that $y'' \geq z_b$: we only need to verify the preconditions of Claim 3.8.5, which follows by (3.80) together with the fact that $h^{(N-1)}_y \geq h^{(N)}_y$ for all y. We now upper bound the rhs of the equation above:

$$\begin{aligned}
& \frac{2(z_a/x)^2-1}{2(z_a/y'')^2-1}\cdot\frac{1}{\widetilde{T}} - \frac{2(z_a/x)^2-1}{2(z_a/z_b)^2-1}\cdot\frac{1}{\widetilde{T}} \\
\leq& \frac{2(z_a/x)^2-1}{\widetilde{T}}\left(1 - \frac{1}{2(z_a/z_b)^2-1}\right) \\
\leq& \frac{2(z_a/x)^2-1}{\widetilde{T}} \\
\leq& \frac{2(z_a/\Delta)^2-1}{\widetilde{T}} \\
=& \frac{O(1)}{\widetilde{T}} && \text{Since } z_a \leq R_{max} = O(1) \text{ and } \Delta = \Omega(1)
\end{aligned}$$

This completes the inductive claim and establishes (3.70) for all $j = 1, \ldots, j^*$.

The only thing we need to verify is that the solution that we presented, satisfies the **non-empty range constraint**. For the sake of this proof, let us define $g_{z_{j^*-1},j^*}$ as follows:

$$g_{z_{j^*-1},j^*} := g_{z_{j^*-1},j^*-1} - \frac{2\left(z_{j^*-1}/x\right)^2-1}{2\left(z_{j^*-1}/z_{j^*-1}\right)^2-1}\cdot\alpha_{j^*-1} = g_{z_j,j} - \frac{2\left(z_{j^*-1}/x\right)^2-1}{1}\cdot\alpha_{j^*-1} \tag{3.86}$$

If s is such that $c_s = z_{j^*-1}$ and $c_{s-1} \neq z_{j^*-1}$, then by the discussion above

$$f_{y,s} \leq g_{y,j^*-1} + X_s. \tag{3.87}$$

and more specifically, when $y = z$ by the assumption we have

$$7\delta_z \leq f_{z,J+1} \leq g_{z,j^*} + O\left(\frac{1}{\sqrt{T}}\right) + 3\delta_z \qquad \text{By (3.78)}$$

which implies

$$g_{z,j^*} \geq 3\delta_z. \tag{3.88}$$

Now, we prove that $g_{z_{j^*},j^*} \geq 0$. The same as the discussion above, if we took N steps on sphere z_{j^*-1} then by (3.67) we have

$$g_{z,j^*} = h_z^{(N)},$$

where h's are the auxiliary variables defined for sphere z_{j^*-1}. Now, we also have

$$\begin{aligned} g_{z,j^*} = h_z^{(N)} &\leq h_z^{(N-1)} && \text{By (3.65)} \\ &\leq h_{z_{j^*-1}}^{(N-1)} && \text{By (3.64)} \end{aligned}$$

On the other hand, we have

$$h_{z_{j^*-1}}^{(N)} = h_{z_{j^*-1}}^{(N-1)} - \frac{2(z_{j^*-1}/x)^2 - 1}{1} \frac{1}{\tilde{T}} = h_{z_{j^*-1}}^{(N-1)} - O\left(\frac{1}{T}\right) \geq h_{z_{j^*-1}}^{(N-1)} - \delta_z$$

Combining these facts we get

$$g_{z_{j^*-1},j^*} = h_{z_{j^*-1}}^{(N)} \geq h_{z_{j^*-1}}^{(N-1)} - \delta_z \geq g_{z,j^*} - \delta_z \geq 2\delta_z$$

where the last inequality is due to (3.88). Also, by the construction of the solution and the fact that the function $\frac{2(z/x)^2-1}{2(z/y)^2-1}$ is increasing in y, we have

$$\begin{aligned} g_{z_{j^*-1},j^*} - g_{z_{j^*},j^*} &\leq \min\left\{\frac{z_{j^*-1}^2 - x^2}{2}, 1 - \frac{x^2}{2}\right\} - \min\left\{\frac{z_{j^*}^2 - x^2}{2}, 1 - \frac{x^2}{2}\right\} \\ &\leq \frac{(\sqrt{2})^2 - \left((1-\delta_z)\sqrt{2}\right)^2}{2} \leq 2\delta_z \end{aligned}$$

which implies that $g_{z_{j^*},j^*} \geq 0$ (the non-empty range constraint in the LP (3.33)).

Now, recalling the values of α_j's, one can see the cost of this solution of LP is equal to $\frac{J(1-10^{-4})}{T}$, which is greater than the optimal solution for the LP, which is a contradiction. So, the claim holds. □

It is important to note that Claim 3.8.6 is not universally true for any shift-invariant kernel, even under natural monotonicity assumptions. An example is presented in Section 3.2 in Figure 3.4. We show, however, that our linear programming formulation is indeed a tight relaxation for a wide class of kernels that includes the Gaussian kernel, the exponential kernel as well as any log-convex kernel.

3.9 Upper bounding LP value

The main result of this section is a proof of Lemma 3.9.1 below:

Lemma 3.9.1. *For every $x \geq 0, y \geq x$ the value of the LP in* (3.33) *(restated below as* (3.89)*) is upper bounded by* 0.1718. *Furthermore, for every $x \in (0, \sqrt{2})$ and $y \geq \sqrt{2}$ the LP value is bounded by $3 - 2\sqrt{2} < 0.1718$, and for every $x \in (0, \sqrt{2})$ and every $y \in [x, \sqrt{2}]$ the LP value is bounded by $\frac{x^2}{2}\left(1 - \frac{x^2}{2}\right)$.*

The main result of this section is an upper bound on the value of the LP (3.89) below. We first derive a dual formulation, then exhibit a feasible dual solution and then verify numerically that the value of dual is bounded by 0.172 for all values of the input parameters x and y^*.

Fix $x \in [0, \sqrt{2}]$. Let $z_1 > z_2 > \ldots > z_I$, denote the distances on the grid, and we define $Z := \{z_1, z_2, \ldots, z_I\}$, we will consider I linear programs, enumerating over all $j^* \in [I]$ such that $z_{j^*} \geq x$.

$$\max_{\alpha \geq 0} \sum_{j=1}^{j^*-1} \alpha_j \tag{3.89}$$

such that :

$$\forall y \in Z : g_{y,1} \leq \min\left\{\frac{y^2 - x^2}{2}, 1 - \frac{x^2}{2}\right\} \qquad (r_{y,0}) \quad \text{Density constraints}$$

$$\forall j \in [j^* - 1], \forall y \in Z \text{ s.t. } y < z_j : \; g_{y,j} \leq g_{z_j,j} \qquad (q_{y,j}) \quad \text{Truncation}$$

$$g_{y,j+1} \leq g_{y,j} - \frac{2\left(z_j/x\right)^2 - 1}{2\left(z_j/y\right)^2 - 1} \cdot \alpha_j \qquad (r_{y,j}) \quad \text{Spherical LSH}$$

$$g_{z_{j^*},j^*} \geq 0 \qquad (\eta)$$

The dual of (3.89) is

$$\min \sum_{y \in Z} \left\{\frac{y^2 - x^2}{2}, 1 - \frac{x^2}{2}\right\} r_{y,0} \tag{3.90}$$

such that :

$$\forall j \in [j^* - 1], y \in Z, y < z_j : r_{y,j-1} - r_{y,j} + q_{y,j} = 0 \qquad (g_{y,j}) \quad \text{Mass transportation}$$

$$\forall j \in [j^* - 1] : r_{z_j,j-1} - \sum_{x \in Z, x < z_j} q_{x,j} = 0 \qquad (g_{z_j,j}) \quad \text{Max tracking}$$

$$\forall y \in Z, y < z_{j^*} : r_{y,j^*-1} = 0 \qquad (g_{y,j^*}) \quad \text{Sink}$$

$$-\eta + r_{z_{j^*},j^*-1} = 0 \qquad (g_{z_{j^*},j^*}) \quad \text{Terminal flow}$$

$$j \in [j^* - 1] : \sum_{y \in Z : y < z_j} \frac{2\left(z_j/x\right)^2 - 1}{2\left(z_j/y\right)^2 - 1} r_{y,j} \geq 1 \qquad (\alpha_j)$$

$$r_{y,j}, q_{y,j} \geq 0$$

$$\eta \geq 0$$

We start by exhibiting a simple feasible solution for the dual that reproduces our result from Section 3.4.

Upper bound of $\frac{x^2}{2} \cdot (1 - \frac{x^2}{2})$ for every x. Let $q_{y,j} = 0$ for all y, j. Let

$$r_{z_{j^*},j} = \left(\frac{x}{y}\right)^2$$

for all $j = 0, 1, \ldots, j^* - 1$ and let $r_{y,j} = 0$ for $y \neq z_{j^*}$ and all y. We let $\eta = r_{z_{j^*}, j^*-1}$. We first verify feasibility. We have for every $j = 1, \ldots, j^* - 1$

$$\begin{aligned}
\sum_{y \in Z : y < z_j} \frac{2\left(z_j/x\right)^2 - 1}{2\left(z_j/y\right)^2 - 1} r_{y,j} &= \frac{2\left(z_{j^*}/x\right)^2 - 1}{2\left(z_{j^*}/y\right)^2 - 1} r_{z_{j^*}, j^*-1} \\
&\geq \frac{2\left(z_{j^*}/x\right)^2}{2\left(z_{j^*}/y\right)^2} r_{z_{j^*}, j^*-1} \\
&= \left(\frac{y}{x}\right)^2 \cdot \left(\frac{x}{y}\right)^2 \\
&= 1,
\end{aligned}$$

where we used the fact that $x \leq y$. We thus have a feasible solution. The value of the solution is

$$\left(\frac{x}{y}\right)^2 \cdot \min\left\{\frac{y^2 - x^2}{2}, 1 - \frac{x^2}{2}\right\} \leq \left(\frac{x}{y}\right)^2 \cdot \min\left\{\frac{y^2 - x^2}{2}, 1 - \frac{x^2}{2}\right\}$$

When $y \geq \sqrt{2}$, we get, $\left(\frac{x}{y}\right)^2 \cdot \left(1 - \frac{x^2}{2}\right)$, which is maximized at $y = \sqrt{2}$ and gives $\left(\frac{x}{\sqrt{2}}\right)^2 \cdot (1 - \frac{x^2}{2})$. Similarly, when $y \leq \sqrt{2}$, we get

$$\left(\frac{x}{y}\right)^2 \cdot \left(\frac{y^2 - x^2}{2}\right) = \frac{x^2}{2} - \frac{x^4}{2y^2},$$

which is again maximized when $y = \sqrt{2}$. Thus, we get that the value of the LP in (3.89) is bounded by

$$\frac{x^2}{2} \cdot (1 - \frac{x^2}{2}).$$

and we obtain the exponent of 0.25. This (almost) recovers the result of Section 3.4 . In what follows we obtain a stronger bound of 0.1718 on the value of the LP in (3.89), obtaining our main result on data-dependent KDE.

Upper bound of 0.1718 **on LP value for all** x**.** We exhibit a feasible solution for the dual in which for every $j < j^*$

$$\begin{aligned}
q_{z_{j+1}, j} &> 0 \\
q_{y,j} &= 0 \text{ for all } y < z_{j+1}
\end{aligned} \tag{3.91}$$

and $q_{y,j^*} = 0$ for all $y < z_{j^*}$. We later show numerically that our dual solution is optimal for the Gaussian kernel.

Simplifying (3.90) under the assumptions from (3.91), we get

$$\min \sum_{y\in Z}\left\{\frac{y^2-x^2}{2}, 1-\frac{x^2}{2}\right\} r_{y,0}$$

such that :

$$\begin{aligned}
\forall j \in [j^*-1], y\in Z, y < z_{j+1} &: r_{y,j-1} - r_{y,j} = 0 && (g_{y,j})\\
\forall j \in [j^*-1] &: r_{z_{j+1},j-1} - r_{z_{j+1},j} + q_{z_{j+1},j} = 0 && (g_{y,j})\\
\forall j \in [j^*-1] &: r_{z_j,j-1} - q_{z_{j+1},j} = 0 && (g_{z_j,j})\\
\forall y \in Z, y < z_{j^*} &: r_{y,j^*-1} = 0 && (g_{y,j^*})\\
&-\eta + r_{z_{j^*},j^*-1} = 0 && (g_{z_{j^*},j^*})\\
j \in [j^*-1] &: \sum_{y\in Z: y<z_j} \frac{2\left(z_j/x\right)^2-1}{2\left(z_j/y\right)^2-1} r_{y,j} \geq 1 && (\alpha_j)\\
&r_{y,j}, q_{y,j} \geq 0\\
&\eta \geq 0
\end{aligned}$$

Eliminating the q variables from the above for simplicity, we get, making the inequality for the (α_j) constraints an equality (recall that we only need to exhibit a dual feasible solution),

$$\min \sum_{y\in Z}\left\{\frac{y^2-x^2}{2}, 1-\frac{x^2}{2}\right\} r_{y,0} \tag{3.92}$$

such that :

$$\begin{aligned}
\forall j \in [j^*-1], y\in Z, y < z_{j+1} &: r_{y,j-1} - r_{y,j} = 0 && (g_{y,j})\\
\forall j \in [j^*-1] &: r_{z_{j+1},j-1} = r_{z_{j+1},j} - r_{z_j,j-1} && (g_{y,j})\\
\forall y \in Z, y < z_{j^*} &: r_{y,j^*-1} = 0 && (g_{y,j^*})\\
&r_{z_{j^*},j^*-1} = \eta && (g_{z_{j^*},j^*})\\
j \in [j^*-1] &: \sum_{y\in Z: y<z_j} \frac{2\left(z_j/x\right)^2-1}{2\left(z_j/y\right)^2-1} r_{y,j} = 1 && (\alpha_j)\\
&r_{y,j} \geq 0\\
&\eta \geq 0
\end{aligned}$$

Defining a dual feasible solution r. We now derive an expression for a feasible solution r. The construction is by **induction**: starting with $j = j^*-1$ as the **base** we define $r_{y,j}$ variables for $y \in Z, y \leq z_j$ that satisfy dual feasibility. The **base** is provided by

$$r_{z_{j^*},j^*-1} = \eta = \left(\frac{2\left(z_{j^*-1}/x\right)^2-1}{2\left(z_{j^*-1}/z_{j^*}\right)^2-1}\right)^{-1}. \tag{3.93}$$

Note that this fully defines $r_{y,j}$ for $j = j^*-1$, since $r_{y,j^*-1} = 0$ for $y < z_{j^*}$.

We now give the **inductive step:** $j \to j-1$. By the inductive hypothesis the variables $r_{y,j}$ that we defined satisfy the (α_j) constraints in the dual (3.92), which means:

$$\sum_{i>j} \frac{2\left(z_j/x\right)^2-1}{2\left(z_j/z_i\right)^2-1} r_{z_i,j} = 1. \tag{3.94}$$

We will define $r_{y,j-1}$ so that

$$\sum_{i>j-1} \frac{2\left(z_{j-1}/x\right)^2-1}{2\left(z_{j-1}/z_i\right)^2-1} r_{z_i,j-1} = 1 \tag{3.95}$$

and at the same time the $(g_{y,j})$ constraints relating $r_{y,j}$ to $r_{y,j-1}$ are satisfied.

By the first constraint in (3.92) we have $r_{z_i,j-1} = r_{z_i,j}$ for all $i > j+1$, since $y < z_{j+1}$ is equivalent to $i > j+1$. By the second constraint in (3.92) we have $r_{z_{j+1},j-1} = r_{z_{j+1},j} - r_{z_j,j-1}$. Putting these two constraints together, we now find $r_{z_j,j-1}$ and therefore $r_{z_{j+1},j-1}$. We rewrite the left hand side of (3.95) as

$$\begin{aligned}
\sum_{i>j-1} \frac{2\left(z_{j-1}/x\right)^2-1}{2\left(z_{j-1}/z_i\right)^2-1} r_{z_i,j-1} &= \frac{2\left(z_{j-1}/x\right)^2-1}{2\left(z_{j-1}/z_j\right)^2-1} r_{z_j,j-1} + \frac{2\left(z_{j-1}/x\right)^2-1}{2\left(z_{j-1}/z_{j+1}\right)^2-1}\left(r_{z_{j+1},j} - r_{z_j,j-1}\right) \\
&\quad + \sum_{i>j+1} \frac{2\left(z_{j-1}/x\right)^2-1}{2\left(z_{j-1}/z_i\right)^2-1} r_{z_i,j} \\
&= \left(\frac{2\left(z_{j-1}/x\right)^2-1}{2\left(z_{j-1}/z_j\right)^2-1} - \frac{2\left(z_{j-1}/x\right)^2-1}{2\left(z_{j-1}/z_{j+1}\right)^2-1}\right) r_{z_j,j-1} \\
&\quad + \sum_{i>j} \frac{2\left(z_{j-1}/x\right)^2-1}{2\left(z_{j-1}/z_i\right)^2-1} r_{z_i,j}
\end{aligned} \tag{3.96}$$

Combining this with (3.95), we thus get that

$$r_{z_j,j-1} = \left(\frac{2\left(z_{j-1}/x\right)^2-1}{2\left(z_{j-1}/z_j\right)^2-1} - \frac{2\left(z_{j-1}/x\right)^2-1}{2\left(z_{j-1}/z_{j+1}\right)^2-1}\right)^{-1} \left(1 - \sum_{i>j} \frac{2\left(z_{j-1}/x\right)^2-1}{2\left(z_{j-1}/z_i\right)^2-1} r_{z_i,j}\right). \tag{3.97}$$

We now show that $r_{z_j,j-1} \geq 0$. The first multiplier in the expression above is non-negative since $\frac{2(z_{j-1}/x)^2-1}{2(z_{j-1}/z)^2-1}$ is increasing in z and $z_j \geq z_{j+1}$. For the second multiplier we have

$$\begin{aligned}
1 - \sum_{i>j} \frac{2\left(z_{j-1}/x\right)^2-1}{2\left(z_{j-1}/z_i\right)^2-1} r_{z_i,j} &= 1 - \sum_{i>j} \frac{2\left(z_{j-1}/x\right)^2-1}{2\left(z_{j-1}/z_i\right)^2-1} r_{z_i,j} \\
&\geq 1 - \sum_{i>j} \frac{2\left(z_j/x\right)^2-1}{2\left(z_j/z_i\right)^2-1} r_{z_i,j} \\
&= 0.
\end{aligned} \tag{3.98}$$

Here the first transition used (3.94) (the inductive hypothesis), and the second transition used the fact that the function $\frac{2(z/x)^2-1}{2(z/y)^2-1}$ is non-increasing in z for $x \leq y$. To summarize, we let $r_{z_j,j-1}$ be defined by (3.97).

Also, we let

$$r_{z_{j+1},j-1} = r_{z_{j+1},j} - r_{z_j,j-1} \tag{3.99}$$

and let $r_{z_i,j-1} = r_{z_i,j}$ for $i > j+1$. We verify numerically that $r_{z_{j+1},j-1} \geq 0$.

Integral equation representation of the dual solution. While we do not use the following in our analysis, it is interesting to note that the dual solution that we propose satisfies an integral equation in the limit as the grid size goes to 0. Let z_j denote a uniform grid with step size $\Delta \to 0$ on the interval $[0, C]$ for some constant $C \geq \sqrt{2}$. We now rewrite (3.97) as

$$\left(\frac{2\left(z_{j-1}/x\right)^2-1}{2\left(z_{j-1}/z_j\right)^2-1} - \frac{2\left(z_{j-1}/x\right)^2-1}{2\left(z_{j-1}/z_{j+1}\right)^2-1}\right) r_{z_j,j-1} = \left(1 - \sum_{i>j} \frac{2\left(z_{j-1}/x\right)^2-1}{2\left(z_{j-1}/z_i\right)^2-1} r_{z_i,j}\right). \tag{3.100}$$

Note by the Mean Value Theorem and the fact that the derivative $\left(\frac{2(z_{j-1}/x)^2-1}{2(z_{j-1}/z)^2-1}\right)'_z$ is Lipschitz within $[z_{j+1}, z_j]$ it follows that

$$\left(\frac{2\left(z_{j-1}/x\right)^2-1}{2\left(z_{j-1}/z_j\right)^2-1} - \frac{2\left(z_{j-1}/x\right)^2-1}{2\left(z_{j-1}/z_{j+1}\right)^2-1}\right) = \left(\frac{2\left(z_{j-1}/x\right)^2-1}{2\left(z_{j-1}/z\right)^2-1}\right)'_z \Bigg|_{z=z_j} \cdot \Delta \cdot (1 + O(\Delta)).$$

We thus have that $r_{z_j,j-1}/\Delta$ converges to the solution $g(y)$ to the following integral equation:

$$\left(\frac{2\left(u/x\right)^2-1}{2\left(u/z\right)^2-1}\right)'_z \Bigg|_{z=u} \cdot g(u) = 1 - \int_{y^*}^{u} \frac{2\left(u/x\right)^2-1}{2\left(u/r\right)^2-1} g(r)\,dr \tag{3.101}$$

The initial condition is a point mass at y^*.

Exact solution to the primal when $z_{j^*} \geq \sqrt{2}$. We note that if $z_{j^*} \geq \sqrt{2}$ an optimal solution to the LP (3.89) is easy to obtain. The reason is that we can simplify the constraints of LP for band z_{j^*} as follows: for all $j \in [j^* - 2]$

$$\begin{aligned} g_{z_{j+2},j+1} &\leq \min\left\{ g_{z_{j+2},j} - \frac{2\left(z_j/x\right)^2-1}{2\left(z_j/z_{j+2}\right)^2-1} \cdot \alpha_j,\ g_{z_{j+1},j} - \frac{2\left(z_j/x\right)^2-1}{2(z_j/z_{j+1})^2-1} \cdot \alpha_j \right\} \\ &\leq g_{z_{j+1},j} - \frac{2\left(z_j/x\right)^2-1}{2(z_j/z_{j+1})^2-1} \cdot \alpha_j \end{aligned}$$

and

$$g_{z_{j^*},j^*} \leq g_{z_{j^*},j^*-1} - \frac{2\left(z_{j^*-1}/x\right)^2-1}{2\left(z_{j^*-1}/z_{j^*}\right)^2-1} \cdot \alpha_{j^*-1}.$$

Now, combining these inequalities with the fact that $g_{z_2,1} \le 1 - \frac{x^2}{2}$, one has

$$\begin{aligned} g_{z_{j^*},j^*} &\le 1 - x^2/2 - \sum_{j=1}^{j^*-1} \frac{2\left(z_j/x\right)^2 - 1}{2\left(z_j/z_{j+1}\right)^2 - 1}\alpha_j \\ &\le 1 - \frac{x^2}{2} - \frac{1}{1+5\delta_z}\sum_{j=1}^{j^*-1}\left(2\left(z_j/x\right)^2 - 1\right)\cdot\alpha_j \\ &\le 1 - \frac{x^2}{2} - \frac{1}{1+5\delta_z}\sum_{j=1}^{j^*-1}\left(2\left(z_{j^*}/x\right)^2 - 1\right)\cdot\alpha_j \\ &\le 1 - \frac{x^2}{2} - \frac{1}{1+5\delta_z}\left(2\left(\sqrt{2}/x\right)^2 - 1\right)\cdot\sum_{j=1}^{j^*-1}\alpha_j, \end{aligned} \tag{3.102}$$

where we used the fact that $2\left(z_j/z_{j+1}\right)^2 - 1 = 2(1+\delta_z)^2 - 1 \le 1 + 5\delta_z$ and the function $2\left(z/x\right)^2 - 1$ is increasing in z.

Letting $\gamma := \sum_{j=1}^{j^*-1}\alpha_j$ denote the LP objective we need to maximize γ subject to $1 - x^2/2 - \frac{1}{1+5\delta_z}\left(2\left(\sqrt{2}/x\right)^2 - 1\right)\cdot\gamma \ge 0$ (the nonempty range LP constraint), where $y^* = z_{j^*}$. The solution is

$$\gamma = (1+5\delta_z)\left(1 - x^2/2\right)\left(2\left(\sqrt{2}/x\right)^2 - 1\right)^{-1}.$$

Finally, one has

$$\max_{x\in[0,\sqrt{2}]}\left(\frac{1 - x^2/2}{2(2/x^2) - 1}\right) = 3 - 2\sqrt{2} \approx 0.171573,$$

which is achieved at $x = \sqrt{4 - 2\sqrt{2}}$. It remains to note that this is achievable by letting $\alpha_{j^*-1} = \gamma$ and letting $\alpha_j = 0$ for $j < j^* - 1$, when $z_{j^*} = \sqrt{2}$.

Numerical verification for $x \in [0,\sqrt{2}], y \in [0,\sqrt{2}]$. Implementing this in Matlab and optimizing over x and j^* (with a uniform grid on $[0,\sqrt{2}]$ consisting of $J = 400$ points) yields the exponent of ≈ 0.1716, achieved at $x \approx 1.0842$ and $z_{j^*} \approx \sqrt{2}$. The Matlab code is given below. Then 0.1718 is an upper-bound on the optimal cost of LP. Moreover, for the analysis if we set $\alpha^* = 0.172$ (as in Section 3.5.2) then $\alpha^*(1 - 10^{-4})$ strictly upper bounds OPT(LP) (this simplifies the notation in other sections).

```
J=400;
vmax=0;
xIdxMax=0;
yIdxMax=0;

for xIdx=5:5:J-5,
for yIdx=xIdx-5:-5:1,
z=sqrt(2)*(J-(1:J))/J;
density=zeros(J);
for j=1:J,
%% density for exp(-x^2/2)
density(j)=min((z(j)^2-z(xIdx)^2)/2, 1-z(xIdx)^2/2);
end;
r=zeros(J);
r(yIdx-1)=((2*(z(yIdx-1)/z(xIdx))^2-1)/(2*(z(yIdx-1)/z(yIdx))^2-1))^(-1);
for j=yIdx-2:-1:1,
coeff=zeros(J);
for i=j:yIdx,
coeff(i)=(2*(z(j)/z(xIdx))^2-1)/(2*(z(j)/z(i))^2-1);
end;
val=0;
for i=j+1:yIdx-1,
val=val+coeff(i+1)*r(i);
end;

r(j)=(coeff(j+1)-coeff(j+2))^(-1)*(1-val);
r(j+1)=r(j+1)-r(j);
end;
val=0;
for i=1:J,
val=val+density(i)*r(i);
end;
if vmax<val
vmax=val;
xIdxMax=xIdx;
yIdxMax=yIdx;
end;
end;
end;

vmax
xIdxMax
yIdxMax
%%%%%%%%%%

Matlab output:

vmax = 0.1716

xIdxMax = 95

yIdxMax = 5
%%%%%%%%%%
```

For other densities replace the density assignment above accordingly. For example, for the $\exp(-||x||_2)$ (exponential kernel, scaled by $\sqrt{2}$ for convenience) set

```
%% density for exp(-|x|/sqrt{2})
density(j)=min((z(j)-z(xIdx))/sqrt(2), 1-z(xIdx)/sqrt(2));
```

and for the $\exp\left(-\sqrt{||x||_2}\right)$ kernel (scaled to $\sqrt{2}$ for convenience) set

```
%% density for exp(-(x/\sqrt{2})^{1/2})
density(j)=min(sqrt(z(j)/sqrt(2))-sqrt(z(xIdx)/sqrt(2)), 1-sqrt(z(xIdx)/sqrt(2)));
```

respectively.

4 Conclusion

The first part of this thesis presented an almost optimal algorithm for spectral sparsification in dynamic streams and shed light on a better understanding of the effective resistance embedding of graphs. Moreover, our result shows a non-trivial and surprising application of locality-sensitive hash functions in this embedding. The unexpectedness of this approach is that we are interested in finding edges with relatively separated endpoints in the effective resistance embedding, but generally, we use LSH to find relatively close pairs in metric spaces. One possible future research direction for this problem would be to find a practical version of this algorithm, improve the poly-logarithmic factors, and ultimately close the poly-logarithmic gap between the upper bounds and lower bounds.

In the second part of this thesis, we presented the first non-trivial single pass algorithm for spanner construction in dynamic streams. Furthermore, this result presented an exciting connection between the effective resistance metric and the shortest path metric of graphs. Proving or refuting Conjecture 2.1.1 is a possible future research direction in this area that can enable us to improve our understanding of this connection. Moreover, we presented results for communication round-stretch trade-offs in the simultaneous communication model. However, the optimal trade-offs are unknown currently and need further research.

In the final part of this section, we first presented a clean and easy-to-analyze algorithm using Euclidean LSH that improved or matched up to poly-logarithmic factors to the best-known results in the literature. Later, using data-dependent LSH algorithms, we improved this result further for a class of kernels, including Gaussian kernels. One major challenge is simplifying the data-dependent approach's cumbersome runtime analysis. Also, leveraging our method for designing novel practical algorithms for this problem is an appealing research direction.

A Supplementary Materials for Chapter 1

We will need Lemma A.0.1 that we use in the correctness proof of our algorithm.

Lemma A.0.1 (Chain of Coarse Sparsifiers [127, 7])**.** *Consider any PSD matrix K with maximum eigenvalue bounded from above by $\lambda_u = 2n$ and minimum nonzero eigenvalue bounded from below by $\lambda_\ell = \frac{1}{8n^2}$. Let $d = \lceil \log_2 \frac{\lambda_u}{\lambda_\ell} \rceil$. For $\ell \in \{0,1,2,\ldots,d\}$, define:*

$$\gamma(\ell) = \frac{\lambda_u}{2^\ell}.$$

So $\gamma(d) \leq \lambda_\ell$, and $\gamma(0) = \lambda_u$. Then the chain of PSD matrices, $[K_0, K_1, \ldots, K_d]$ with $K_\ell = K + \gamma(\ell) I$ satisfies the following relations:

1. $K \preceq_r K_d \preceq_r 2 \cdot K$,
2. $K_\ell \preceq K_{\ell-1} \preceq 2 \cdot K_\ell$ *for all* $\ell \in \{1,\ldots,d\}$,
3. $K_0 \preceq 2 \cdot \gamma(0) \cdot I \preceq 2 \cdot K_0$.

We will need Theorem A.0.1 that we use in the proof of correctness of the main algorithm. It is well known that by sampling the edges of B according to their effective resistance, it is possible to obtain a weighted edge vertex incident matrix $\widetilde{B}$ such that $(1-\epsilon)B^\top B \preceq \widetilde{B}^\top \widetilde{B} \preceq (1+\epsilon)B^\top B$ with high probability (see Lemma A.0.1).

Theorem A.0.1 (Spectral Approximation via Effective Resistance Sampling [30])**.** *Let $B \in \mathbb{R}^{\binom{n}{2} \times n}$, $K = B^\top B$, and let $\widetilde{\tau}$ be a vector of leverage score overestimates for B's rows, i.e. $\widetilde{\tau}_y \geq \mathbf{b}_y^\top K^+ \mathbf{b}_y$ for all $y \in [m]$. For $\epsilon \in (0,1)$ and fixed constant c, define the sampling probability for row $\mathbf{b}_y$ to be $p_y = \min\{1, c \cdot \epsilon^{-2} \log n \cdot \widetilde{\tau}_y\}$. Define a diagonal sampling matrix W with $W(y,y) = \frac{1}{p_y}$ with probability p_y and $W(y,y) = 0$ otherwise. With high probability, $\widetilde{K} = B^\top W B \approx_\epsilon K$. Furthermore W has $O(||\widetilde{\tau}||_1 \cdot \epsilon^{-2} \log n)$ non-zeros with high probability.*

Lemma A.0.2 (ℓ_2 Heavy Hitters)**.** *For any $\eta > 0$, there is a decoding algorithm denoted by* HeavyHitter *and a distribution on matrices S^h in $\mathbb{R}^{O(\eta^{-2}\,\mathrm{polylog}(N)) \times N}$ such that, for any $x \in \mathbb{R}^N$, given $S^h x$, the algorithm* HeavyHitter$(S^h x, \eta)$ *returns a list $F \subseteq [N]$ such that $|F| = O(\eta^{-2}\,\mathrm{polylog}(N))$ with probability $1 - \frac{1}{poly(N)}$ over the choice of S^h one has*

(1) *for every* $i \in [N]$ *such that* $|x_i| \geq \eta ||x||_2$ *one has* $i \in F$*;*

(2) *for every* $i \in F$ *one has* $|x_i| \geq (\eta/2)||x||_2$.

The sketch $S^h x$ *can be maintained and decoded in* $O(\eta^{-2}\,\mathrm{polylog}(N))$ *time and space.*

Lemma A.0.3 (Binary Johnson-Lindenstrauss Lemma [128])**.** *Let* P *be an arbitrary set of points in* $\mathbb{R}^d$*, represented by a* $d \times n$ *matrix* A*, such that the* j^{th} *point is* $A\chi_j$*. Given* $\epsilon,\ \beta > 0$ *and*

$$q \geq \frac{4+2\beta}{\epsilon^2/2 - \epsilon^3/3} \log n.$$

Let Q *be a random* $q \times d$ *matrix* $(q_{ij})_{ij}$ *where* q_{ij}*'s are independent identically distributed variables taking* 1 *and* -1 *each with probability* $1/2$*. Then, if* $M = \frac{1}{\sqrt{q}} QA$*, then with probability at least* $1 - n^{-\beta}$*, for all* $u, v \in [n]$

$$(1-\epsilon)||A\chi_u - A\chi_v||_2^2 \leq ||M\chi_u - M\chi_v||_2^2 \leq (1+\epsilon)||A\chi_u - A\chi_v||_2^2$$

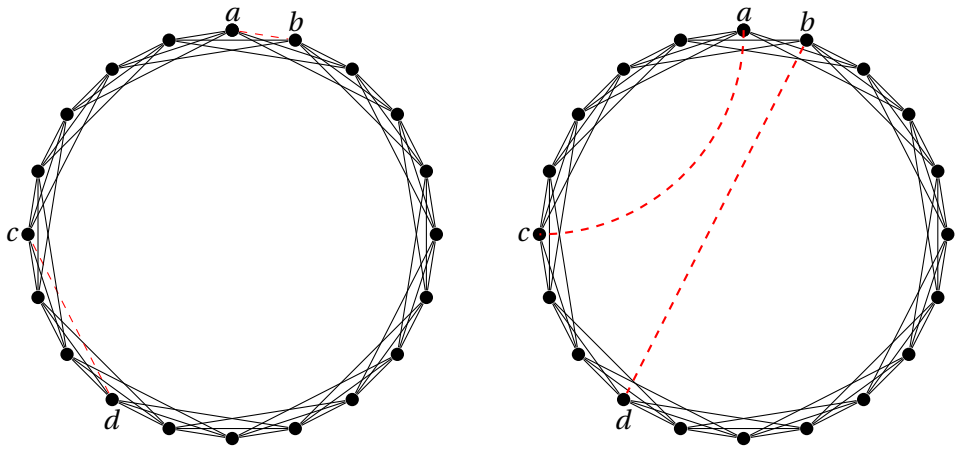
B Supplementary Materials for Chapter 2

B.1 Conjectured hard input distribution

Let $\pi : [n] \to [n]$ be a uniformly random permutation. Let $d > 1$ be an integer parameter. Define the distribution $\mathscr{D}'$ on graphs $G = (V, E)$, $V = [n]$ as follows. For every pair $(i, j) \in [n]$ such that $\|\pi(i) - \pi(j)\|_\circ \leq d$ include an edge (i, j) in E with probability $1/2$, where $\|i - j\|_\circ$ is the circular distance on a cycle of length n. Define the distribution $\mathscr{D}$ on graphs $G = (V, E)$ as follows. First sample $G' = (V, E') \sim \mathscr{D}'$, and pick two edges $(a, b), (c, d) \sim \text{Unif}(E')$ independently without replacement, and let

$$E = (E' \cup \{(a, c), (b, d)\}) \setminus \{(a, b), (c, d)\}.$$

Let $G = (V, E)$ be a sample from $\mathscr{D}$. Note that with constant probability over the choice of $G \sim \mathscr{D}$ one has that the distance between G from a to c in $E \setminus \{(a, c), (b, d)\}$ is $\Omega(n/d)$ and the distance between c and d in $E \setminus \{(a, c), (b, d)\}$ is $\Omega(n/d)$ (see Figure B.1 for an illustration). Thus, every k-spanner with $k \ll n/d$ must contain both of these edges. We conjecture that recovering these edges from a linear sketch of the input graph G sampled from $\mathscr{D}$ requires $n^{1+\Omega(1)}$ space when $d = n^{1/3+\Omega(1)}$. Note that the diameter of the graph is (up to polylogarithmic factors) equals n/d, and hence this would in particular imply that obtaining an $n^{2/3-\Omega(1)}$ spanner using a linear sketch requires $n^{1+\Omega(1)}$ bits of space, and therefore imply Conjecture 2.1.1.



(a) Graph G' (b) Graph G

Figure B.1: Illustration of the conjectured hard input distribution

B.2 Omitted Proofs

B.2.1 Omitted proofs from Section 2.2

Proof of Lemma 2.2.1. First, we start by the following well-known fact about ℓ_0-sampling sketching algorithms.

Fact B.2.1 (See e.g., [25, 129])**.** *For any vector $\vec{a} \in \mathbb{R}^n$ that*

- *receives coordinate updates in a dynamic stream,*
- *and each entry is bounded by $O(poly(n))$,*

one can design an ℓ_0-sampler procedure, which succeeds with high probability, by storing a vector $\vec{b} \in \mathbb{R}^{\text{polylog}(n)}$, where

- *(Bounded entries) each entry of $\vec{b}$ is bounded by $O(poly(n))$,*
- *(Linearity) there exists a matrix Π (called* sketching matrix*) such that $\vec{b} = \Pi \cdot \vec{a}$.*

Now, we use Fact B.2.1 to prepare a data structure for ℓ_0 sampling of A_i for each $i \in [r]$.[1] Thus, we are going to have r vectors, $\vec{b}_1, \ldots, \vec{b}_r$, where for each $i \in [r]$ the entries of $\vec{b}_i$ are bounded by $O(\text{poly}(n))$. However, our space is limited to $s \cdot \text{polylog}\, n$, so we cannot store these vectors. Define $\vec{b}$ as concatenation of $\vec{b}_1, \ldots, \vec{b}_r$. At this point, the reader should note that by assumption $|\mathcal{I}| \leq s$, which implies that at most $s \cdot \text{polylog}(n)$ of entries of $\vec{b}$ are non-zero at the end of the stream.[2] Now, we need to prepare a $\widetilde{O}(s)$-sparse recovery primitive for $\vec{b}$. We apply the following well-known fact about sparse recovery sketching algorithms.

[1] Note that we do not need to sample uniformly over the non-zero entries, and just recovering a non-zero element is enough for our purpose, however, we still use a ℓ_0-sampling procedure.

[2] However, at some time during the stream, you may have more than $s \cdot \text{polylog}(n)$ non-zeroes, so it is not possible to store $\vec{b}$ explicitly.

Fact B.2.2 (Sparse recovery)**.** *For any vector $\vec{b} \in \mathbb{R}^n$ that*

- *receives coordinate updates in a dynamic stream,*
- *and each entry is bounded by $O(poly(n))$,*

one can design a s-sparse recovery sketching procedure, by storing a vector $\vec{w} \in \mathbb{R}^{s \cdot \text{polylog}(n)}$, where

- *(Bounded entries) each entry of $\vec{w}$ is bounded by $O(poly(n))$,*
- *(Linearity) there exists a matrix Π (called* sketching matrix*), where $\vec{w} = \Pi \cdot \vec{b}$,*

such that if $\text{Support}(\vec{b}) \leq s$*, then it can recover all non-zero entries of $\vec{b}$.*

Using Fact B.2.2, we can store a sketch of $\vec{b}$, in $\widetilde{O}(s)$ bits of space and recover the non-zero entries of $\vec{b}$ at the end of the stream, which in turn recovers a non-zero element from each A_i, in $\vec{v}$. In other words, one can see this procedure as the following linear operation

$$\vec{w} = \Pi_1 \Pi_2 \vec{v}$$

where matrix $\Pi_2 \in \mathbb{R}^{r \cdot \text{polylog}(n) \times n}$ is in charge of ℓ_0 sampling for each A_i and concatenation of vectors, and $\Pi_1 \in \mathbb{R}^{(s \cdot \text{polylog}(n)) \times (r \cdot \text{polylog}(n))}$ is responsible for the sparse recovery procedure.

□

Proof of Lemma 2.2.2. Let vector $\vec{b}$ be an indicator vector for edges in $A \times B$, i.e., each entry corresponds to a pair of vertices in $A \times B$ and is 1 if the edge is in the graph, and is 0 otherwise. Now, by applying Fact B.2.1 to this vector, one can recover an edge using space $O(\text{polylog}(n))$. Also, using Fact B.2.2 with $s = m$, we can recover all m edges using space $m \cdot \text{polylog}(n)$. □

B.2.2 Omitted proofs from Section 2.4

Proof of Theorem 2.4.1. Consider an edge $(u, v) \in E$. Similarly to the proof of Theorem 2.1.1, fix $s := d_{\hat{H}}(u, v)$, and $A_i := \{z \in V \mid d_{\hat{H}}(v, z) = i\}$ for $i \in [0, s-1]$ be all the vertices at distance i from v in $\hat{H}$. Set $A_s := \{z \in V \mid d_{\hat{H}}(v, z) \geq s\}$ to be all the vertices at distance at least s from v. In addition set $W_i^H = w_H(A_i \times A_{i+1})$ and $W_i^G = w_G(A_i \times A_{i+1})$. Also recall that $W_{-1}^H = W_{-1}^G = W_s^H = W_s^G = 0$.

We will follow steps similar to those in Claim 2.4.2. Set

$$\alpha = \Theta(\varepsilon \log n) \text{ such that } \alpha \geq 10 \log_{\frac{1}{6\epsilon}} \frac{2s}{\alpha}, \tag{B.1}$$

and $I = \left\{i \in [0, s-1] \mid W_i^G \leq \frac{\alpha m}{s}\right\}$. It holds that $|I| \geq \left(1 - \frac{1}{\alpha}\right) s + 1$, as otherwise there are more than $\frac{s}{\alpha}$ indices i for which $W_i^G > \frac{\alpha m}{s}$, implying $\sum_i W_i^G > m$, a contradiction, since $\{W_i^G\}_i$ represent the number of

elements in disjoint sets of edges. Set

$$\widetilde{I} = \left\{ i \mid \text{such that } \forall\, j, |i-j| \leq \frac{\alpha}{10} \text{ it holds that } j \in I \right\}.$$

Then there are less than $\frac{s}{\alpha} \cdot \frac{2\alpha}{10} < \frac{s}{2}$ indices out of $\widetilde{I}$, implying

$$\left|\widetilde{I}\right| \geq \frac{s}{2}. \tag{B.2}$$

For any index $i \in \widetilde{I}$ and any index $j \in \left[i - \frac{\alpha}{10}, i + \frac{\alpha}{10} - 1\right]$, by Claim 2.4.1,

$$W_{i-1}^H + W_i^H + W_{i+1}^H \geq \frac{1}{\epsilon}\left(W_i^H - W_i^G\right) \geq \frac{1}{\epsilon}\left(W_i^H - \frac{\alpha m}{s}\right).$$

Assume for contradiction that $W_i^H > 2 \cdot \frac{\alpha m}{s}$. Then,

$$W_{i-1}^H + W_i^H + W_{i+1}^H > \frac{1}{\epsilon}\left(\frac{\alpha m}{s} - \frac{\alpha m}{s}\right) = \frac{1}{\epsilon} \cdot \frac{\alpha m}{s}.$$

Let $i_1 \in \{i-1, i, i+1\}$ such that $W_{i_1}^H \geq \frac{1}{3\epsilon} \cdot \frac{\alpha m}{s} \geq \frac{1}{6\epsilon} \cdot \frac{\alpha m}{s}$. Using the same argument,

$$W_{i_1-1}^H + W_{i_1}^H + W_{i_1+1}^H \geq \frac{1}{\epsilon}\left(W_{i_1}^H - \frac{\alpha m}{s}\right) > \frac{1}{2\epsilon} \cdot \frac{1}{6\epsilon} \cdot \frac{\alpha m}{s}.$$

Choose $i_2 \in \{i_1 - 1, i_1, i_1 + 1\}$ such that $W_{i_2}^H > \frac{1}{(6\epsilon)^2} \cdot \frac{\alpha m}{s}$. As $i \in \widetilde{I}$, we can continue this process for $\frac{\alpha}{10}$ steps, where in the j step we have $W_{i_j}^H > \frac{1}{(6\epsilon)^j} \cdot \frac{\alpha m}{s}$. In particular

$$W_{i_{\frac{\alpha}{10}}}^H > (6\epsilon)^{-\frac{\alpha}{10}} \frac{\alpha m}{s} \geq 2m,$$

a contradiction, as H is an $(1 \pm \epsilon)$-spectral sparsifier of the unweighted graph G, where the maximal size of a cut is m. We conclude that for every $i \in \widetilde{I}$ it holds that $W_i^H \leq 2 \cdot \frac{\alpha m}{s}$. It follows that

$$\begin{aligned}
\sum_{i=0}^{s-1} \frac{1}{W_i^H} &\geq \left|\widetilde{I}\right| \cdot \frac{s}{2\alpha m} \\
&\geq \frac{s^2}{4\alpha m} && \text{By (B.2)} \\
&= \widetilde{\Omega}\left(\frac{s^2}{m}\right) && \text{By setting of } \alpha \text{ in (B.1)}
\end{aligned} \tag{B.3}$$

Construct an auxiliary graph H' from H, by contracting all the vertices inside each set A_i, and keeping multiple edges. Note that by this operation, the effective resistance between u and v can only decrease. The graph H' is a path graph consisting of s vertices, where the conductance between the i'th vertex to

the $i+1$'th is W_i^H. We conclude

$$\begin{aligned}
(1+\epsilon)R_{u,v}^G &\geq R_{u,v}^H && \text{By Fact 2.2.2} \\
&\geq R_{u,v}^{H'} && \text{As explained above} \\
&= \sum_{i=0}^{s-1} \frac{1}{W_i^H} && \text{Since } H' \text{ is a path graph} \\
&= \widetilde{\Omega}\left(\frac{s^2}{m}\right) && \text{By (B.3)} \qquad \text{(B.4)}
\end{aligned}$$

As u, v are neighbors in the unweighted graph G, it necessarily holds that $R_{u,v}^G \leq 1$, implying that $s = \widetilde{O}(\sqrt{m})$. □

C Supplementary Materials for Chapter 3

C.1 Omitted proofs from Section 3.3

Proof of Lemma 3.3.3: Let $x' := ||\mathbf{q}' - \mathbf{p}||$. If we consider the plane containing q, p, o, then q' also belongs to this plane, since this plane contains $\mathbf{q}, o$.[1] Then, without loss of generality we can assume that we are working on $\mathbb{R}^2$, where $o = (0,0), \mathbf{q} = (R_1, 0), \mathbf{q}' = (R_2, 0)$. Let $\mathbf{p} = (\alpha, \beta)$ such that

$$\begin{aligned} x^2 &= (\alpha - R_1)^2 + \beta^2 = \alpha^2 + \beta^2 - 2\alpha R_1 + R_1^2 \\ R_2^2 &= \alpha^2 + \beta^2 \\ x'^2 &= (\alpha - R_2)^2 + \beta^2 = \alpha^2 + \beta^2 - 2\alpha R_2 + R_2^2 \end{aligned}$$

Therefore, one has

$$x^2 = R_2^2 - 2\alpha R_1 + R_1^2.$$

Thus,

$$\begin{aligned} x'^2 &= R_2^2 - 2\alpha R_2 + R_2^2 \\ &= 2R_2(R_2 - \alpha) \\ &= 2R_2\left(R_2 - \frac{R_2^2 + R_1^2 - x^2}{2R_1}\right) \\ &= \frac{R_2}{R_1}\left(x^2 - (R_1 - R_2)^2\right), \end{aligned}$$

which proves the claim. One should note that the claim holds for both $R_1 \geq R_2$ and $R_1 < R_2$. □

Proof of Claim 3.3.1: Now let $\mathbf{q}'$ be the projection of the query on the sphere and let $\mathbf{q}''$ be the antipodal

[1] The cases when $\mathbf{q}' = \mathbf{p}$ or $||\mathbf{p} - \mathbf{q}|| = R_1 + R_2$ are the cases when the plane is not unique, but the reader should note that these cases correspond to $x' = 0, x' = 2R_2$ cases, which are trivial.

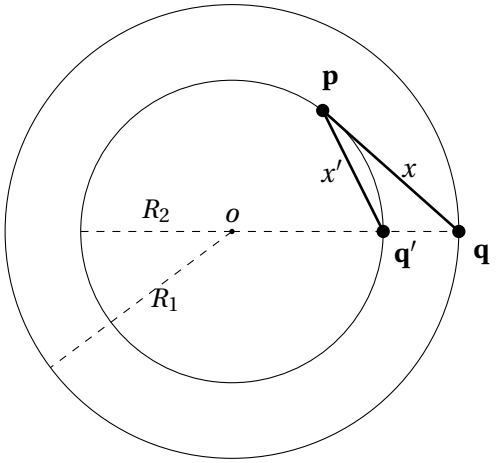


Figure C.1: Illustration of $x' = \textsc{Project}(x, R_1, R_2)$

point of $\mathbf{q}'$ on this sphere (see Figure C.2). By Definition 3.3.3, we have

$$\left|\left\{\mathbf{u} \in P : ||\mathbf{u} - \mathbf{q}'|| \leq r(\sqrt{2} - \gamma)\right\}\right| \leq \tau \cdot |P|$$

and

$$\left|\left\{\mathbf{u} \in P : ||\mathbf{u} - \mathbf{q}''|| \leq r(\sqrt{2} - \gamma)\right\}\right| \leq \tau \cdot |P|. \tag{C.1}$$

On the other hand, in Figure C.2 let a, c be points at distances $r(\sqrt{2}-\gamma)$ and $r\sqrt{2}$ respectively from $\mathbf{q}'$ and $\mathbf{d}$ be a point at distance $r(\sqrt{2}-\gamma)$ from $\mathbf{q}''$. Then by Pythagoras theorem, we have

$$||\mathbf{q}' - \mathbf{d}||^2 + |\mathbf{q}'' - \mathbf{d}||^2 = ||\mathbf{q}' - \mathbf{q}''||^2,$$

which implies

$$||\mathbf{q}' - \mathbf{d}|| = r\sqrt{4 - \left(\sqrt{2} - \gamma\right)^2} = r\sqrt{2 - \gamma^2 + 2\sqrt{2}\gamma},$$

since $||\mathbf{q}'' - \mathbf{d}|| = ||\mathbf{q}' - \mathbf{a}||$. Therefore, we have the following

$$\left|\left\{\mathbf{u} \in P : ||\mathbf{u} - \mathbf{q}''|| \leq r\left(\sqrt{2} - \gamma\right)\right\}\right| = \left|\left\{\mathbf{u} \in P : ||\mathbf{u} - \mathbf{q}'|| \geq r\left(\sqrt{2 - \gamma^2 + 2\sqrt{2}\gamma}\right)\right\}\right| \leq \tau \cdot |P|.$$

So one has

$$\left|\left\{\mathbf{u} \in P : ||\mathbf{u} - \mathbf{q}'|| \in \left(r\left(\sqrt{2} - \gamma\right), r \cdot \sqrt{2 - \gamma^2 + 2\sqrt{2}\gamma}\right)\right\}\right| \geq (1 - 2\tau) \cdot |P|.$$

On the other hand, we have

$$\left(r\left(\sqrt{2} - \gamma\right), r \cdot \sqrt{2 - \gamma^2 + 2\sqrt{2}\gamma}\right) \subseteq \left(r\left(\sqrt{2} - \gamma\right), r\left(\sqrt{2} + \gamma\right)\right),$$

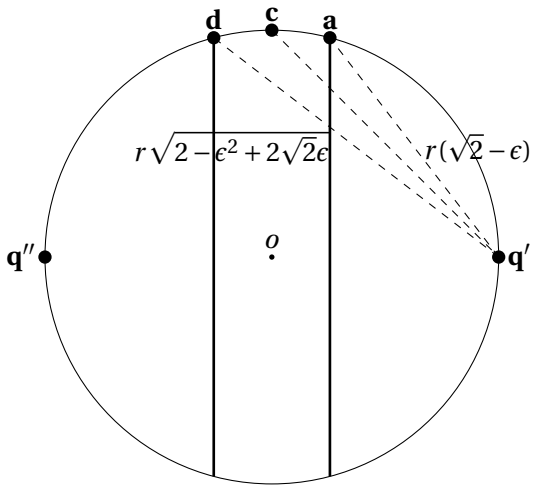


Figure C.2: ϵ-neighborhood of orthogonal points in a sphere of radius r

and hence

$$\left|\left\{\mathbf{u}\in P : ||\mathbf{u}-\mathbf{q}'||\in\left(r\left(\sqrt{2}-\gamma\right), r\cdot\left(\sqrt{2}+\gamma\right)\right)\right\}\right| \geq (1-2\tau)\cdot|P|.$$

which proves the second part of the claim. Now, using (C.1) we have

$$\left|\left\{\mathbf{u}\in P : ||\mathbf{u}-\mathbf{q}'||\leq r(\sqrt{2}-\gamma)\right\}\right| \leq \frac{\tau}{1-2\tau}\cdot\left|\left\{\mathbf{u}\in P : ||\mathbf{u}-\mathbf{q}'||\in\left(r\left(\sqrt{2}-\gamma\right), r\cdot\left(\sqrt{2}+\gamma\right)\right)\right\}\right|.$$

which proves the first part of the claim. □

C.2 Pseudo-random data sets via Ball carving

In this section we provide a simple self-contained proof of the claim that one can efficiently (near-linear time) detect and remove a dense ball on the sphere when it exists that avoids invoking VC-dimension arguments as in [12].

Lemma C.2.1. *There is a randomized procedure* $\textsc{Certify}_{\epsilon,\tau,\delta}(P)$ *that given a set* $P\subset\mathscr{S}^{d-1}$ *and parameters* $\epsilon,\tau,\delta\in(0,\frac{1}{3})$*, runs in time* $O(\frac{d}{\epsilon^2\tau}\log(2n/\delta)\cdot n)$ *and with probability* $1-\delta$

1. *either returns a point* $p^*\in P$ *such that* $|B(p^*,\sqrt{2(1-\epsilon^2)})\cap P|\geq\Omega(\epsilon^2)\cdot\tau|P|$
2. *or certifies that the set* P *is* (ϵ',τ)-pseudo-random *with* $\epsilon'=\sqrt{2}(1-\sqrt{1-2\epsilon})=\Theta(\epsilon)$.

The partitioning procedure is based on the following lemma adapted from [12] showing that in any set that contains a dense ball on the unit-sphere, one can find a point in the dataset that captures a large fraction of the points in the dense ball.

Lemma C.2.2 (Certificate)**.** *Let* $S'\subset\mathscr{S}^{d-1}$ *and* $x^*\in\mathscr{S}^{d-1}$ *such that for* $\epsilon\in(0,\frac{1}{3})$ *and all* $x\in S'$*,* $\|x^*-x\|\leq$

Algorithm 12 CERTIFY$_{\epsilon,\tau,\delta}(P)$

Input: parameters $\epsilon,\tau,\delta \in (0,\frac{1}{3})$ and $P \subset \mathscr{S}^{d-1}$.
$\zeta \leftarrow \frac{1}{4}$, $m \leftarrow \lceil \frac{48}{\epsilon^2\tau}\log(2n/\delta)\rceil$
$Q_m \leftarrow \{m$ uniform random points with replacement from $P\}$ ▷ Sub-sampling
$p^* \leftarrow \arg\max_{p\in P}\{|B(p,\sqrt{2(1-\epsilon^2)})\cap Q_m|\}$
if $|B(p^*,\sqrt{2(1-\epsilon^2)})\cap Q_m| \geq (1-\zeta)(3\epsilon^2)\tau\cdot m$ **then**
    return $p^* \in P$ ▷ Center for a "dense ball" is found.
else
    return $\perp$. ▷ Set P is $(\Theta(\epsilon),\tau)$-pseudo random

$\sqrt{2(1-2\epsilon)}$. *There exists a point* $x_0 \in S'$ *such that*

$$\left|\left\{x \in S' : \|x - x_0\| \leq \sqrt{2(1-\epsilon^2)}\right\}\right| \geq (3\epsilon^2)\cdot|S|'. \tag{C.2}$$

The contrapositive is that if no balls of a certain radius and density exist with points of the dataset as centers, then the dataset is *pseudo-random* with appropriate constants. We can use this lemma to show that either the data set is pseudo-random or we can always find a dense ball and decrease the size of the remaining data set by a non-trivial factor. The issue that is left to discuss is efficiency of the process.

By repeatedly applying the lemma and by stopping only when $|P| \leq \frac{1}{\tau}$ we can decompose any set on the sphere in at most $T = O(\frac{\log|P|}{\epsilon^2\tau})$ "dense balls" and a pseudo-random remainder. Let $\chi \in (0,1)$ be a bound on the failure probability and set $\delta = \chi/T$, then this can be done in time

$$O\left(\frac{d}{\epsilon^4\tau^2}\log\left(2n\log(n)/\epsilon^2\tau\chi\right)\log(n)\cdot n\right). \tag{C.3}$$

For any point $p \in P$ let $B_p := B(p,\sqrt{2(1-\epsilon^2)})\cap P$ and $\mathscr{B} := \{B_p : p \in P \wedge |B_p| \geq 3\epsilon^2\tau\}$. If Q_m is a random sample of m points from P with replacement, then by Chernoff bounds $\forall B \in \mathscr{B}$ we get $\mathbb{P}\left[\left||B\cap Q_m| - \frac{|B|}{|P|}m\right| \geq \zeta\frac{|B|}{|P|}m\right] \leq 2e^{-\frac{\zeta^2}{3}\frac{|B|}{|P|}m}$. Setting $\zeta = \frac{1}{4}$ and $m \geq \frac{48}{\epsilon^2\tau}\log(2n/\delta)$ and taking union bound over at most $|\mathscr{B}| \leq |P|$ events, we get that with probability at least $1-\chi$ for all $B \in \mathscr{B}$ we have $\frac{3}{4}\frac{|B|}{|P|} \leq \frac{|B\cap Q_m|}{m} \leq \frac{5}{4}\frac{|B|}{|P|}$. Conditional on the above event we have that:

- If there exists $p^* \in P$ such that $|B_{p^*}\cap Q_m| \geq \frac{3}{4}(3\epsilon^2)\tau m$, then $|B_{p^*}| \geq \frac{9}{16}(3\epsilon^2)\tau\cdot|P|$.
- If for all $p \in P$, $|B_p\cap Q_m| < (1-\zeta)(3\epsilon^2)\tau m$ then $|B_p| < 3\epsilon^2\tau$ for all $p \in P$ and by Lemma C.2.2 the set P is (ϵ',τ)-pseudo random, with $\epsilon' = \sqrt{2}(1-\sqrt{1-2\epsilon}) \Rightarrow \epsilon = \frac{\epsilon'}{\sqrt{2}}(1-\frac{\epsilon'}{2\sqrt{2}})$ as $\sqrt{2(1-2\epsilon)} = \sqrt{2(1-\sqrt{2}\epsilon'(1-\frac{\epsilon'}{2\sqrt{2}}))} = \sqrt{(\sqrt{2}-\epsilon')^2} = \sqrt{2}-\epsilon'$

This shows correctness of the Procedure CERTIFY$_{\epsilon,\tau,\delta}$ (Algorithm 12). The overall cost of this procedure is $O(dmn) = O(\frac{d\log(2n/\chi)}{\epsilon^2\tau}n)$ dominated by the cost of finding the ball of radius $\sqrt{2(1-\epsilon^2)}$ centered at one of the points in P with the most number of points in Q_m. □

To prove Lemma C.2.2 we are going to use the following simple lemma.

Proposition C.2.1 ([12]). *For any set $S \subset \mathscr{S}^{d-1}$ such that there exists $c \in \mathscr{S}^{d-1}$, $\|c-x\| \le r_\epsilon = \sqrt{2(1-2\epsilon)}$ for all $x \in S$,*

$$\frac{1}{|S|^2}\sum_{x,y\in S}\langle x,y\rangle \ge \left(1-\frac{r_\epsilon^2}{2}\right)^2 = 4\epsilon^2 \tag{C.4}$$

Proof. Given $x, c \in \mathscr{S}^{d-1}$, $\|x-c\| \le r_\epsilon \Rightarrow \langle x,c\rangle \ge 1-\frac{r_\epsilon^2}{2}$. Thus,

$$\sum_{i,j\in S}\langle x_i,x_j\rangle = \|\sum_{x\in S}x\|^2\|c\|^2 \ge \left|\sum_{x\in S}\langle x,c\rangle\right|^2 \ge (1-\frac{r_\epsilon^2}{2})^2|S|^2 \tag{C.5}$$

The proof is concluded by substituting $r_\epsilon = \sqrt{2(1-2\epsilon)}$. □

Proof of Lemma C.2.2. We proceed with a proof by contradiction. Assuming that the statement is not true, then

$$\forall y \in S,\ \left|\left\{x \in S : \|x-y\| \le \sqrt{2(1-\epsilon^2)}\right\}\right| < (3\epsilon^2)\cdot|S| \tag{C.6}$$

Moreover, $\|x-y\| > \sqrt{2(1-\epsilon^2)} \Rightarrow \langle x,y\rangle < \epsilon^2$. Therefore, we get:

$$\frac{1}{|S|^2}\sum_{x,y\in S}\langle x,y\rangle < \left(1-3\epsilon^2\right)\epsilon^2 + 3\epsilon^2\cdot 1 = (4-3\epsilon^2)\epsilon^2 < 4\epsilon^2 \tag{C.7}$$

Using Proposition 1 and the hypothesis we arrive at a contradiction. □

C.3 Correctness proof of the data-dependent algorithm

In this section we present the outer algorithms for our approach. The procedure is quite routine and similar to Section 3.4. First, in Algorithm 13 we present the outer procedure of preprocessing phase. In Algorithm 13 for any $x \in \{\delta_x, 2\delta_x, 3\delta_x, \dots \delta_x\lfloor\frac{\sqrt{2}}{\delta_x}\rfloor\}$, we sample the data set with probability $\min\left\{\frac{1}{n}\left(\frac{1}{\mu}\right)^{1-x^2/2}, 1\right\}$, and then using Algorithm 7, we prepare a data structure that after receiving the query, one can recover any point that is present in the sample and has distance $[x-\delta_x, x)$ from the query using Algorithm 14, with probability 0.8 (see Lemma C.3.1 below).

Lemma C.3.1. *Under Assumption 3.5.1, if $\mathscr{T} = \textsc{PreProcess}(P, x, \mu)$, then for every point $\mathbf{p} \in P$ such that $\mathbf{p} \in v_0.P$, where v_0 is the root of tree $\mathscr{T}$ and $||\mathbf{q}-\mathbf{p}|| \le x$, one has $p \in \textsc{Query}(\mathbf{q}, \mathscr{T}, x)$ with probability at least* 0.8.

Proof. By Corollary 3.5.1, if $\mathscr{H}$ is a (α, x, μ)-AI hash family then for any point $\mathbf{p}$ such that $||\mathbf{p}-\mathbf{q}|| \le x$

$$\mathop{\mathbb{P}}_{h\sim\mathscr{H}}[h(\mathbf{q}) = h(\mathbf{p})] \ge \mu^\alpha$$

Now, noting the number of repetitions of the Andoni-Indyk LSH round, i.e., setting of $K_1 = 100\left(\frac{1}{\mu}\right)^\alpha$ (see line 6 of Algorithm 7), with probability at least 0.9 we know that there exists a hash bucket that both query

and point $\mathbf{p}$ are hashed. Now, we prove by induction on depth of the tree, that if $\mathbf{p}$ belongs to the dataset of root of any tree $\mathcal{T}'$ then $\textsc{Query}(\mathbf{q}, \mathcal{T}', x)$ recovers it with probability at least 0.9.

Base: If the depth of $\mathcal{T}'$ is 1, then $\mathbf{p} \in P_x$ by line 13 of Algorithm 10.
Inductive step: Suppose that $\mathbf{p} \in v.P$ such that v is the root of $\mathcal{T}'$. One should note that v is a pseudo-random sphere. Also, suppose that $\mathbf{q}$ is at distance R_2 from the center of this sphere. Then let $x' := \textsc{Project}(x + \Delta, R_2, R)$ and let x'' be the smallest element in the grid W which is not less than x'. Let $N := \left\lceil \frac{100}{G(x''/R, \eta)} \right\rceil$. Then, by Algorithm 8 we know that v has N children $u_1, u_2, \ldots, u_N$ such that $u_j.x = x''$ for all $j \in [N]$. If $\mathbf{p} \in u_j.P$ for some j then $\mathbf{p}$ will appear in exactly one of the children of u_j, we call this node $u_j(\mathbf{p})$, and if $\mathbf{p} \notin u_j.P$ then $u_j(\mathbf{p}) = \perp$. Let $\mathbf{q}'$ be the projection of $\mathbf{q}$ on the sphere. Now, note that for any $j \in [N]$, if w is a child of u_j then

$$
\begin{aligned}
&\mathbb{P}\left[\textsc{Query}(\mathbf{q}, \mathcal{T}_{u_j(\mathbf{p})}, x) \text{ will be called and } \mathbf{p} \in u_j(\mathbf{p}).P \text{ and } \mathbf{p} \in \textsc{Query}(\mathbf{q}, \mathcal{T}_{u_j(\mathbf{p})}, x)\right] \\
&= \mathbb{P}\left[\left\langle u_j.g, \frac{\mathbf{q}-o}{||\mathbf{q}-o||}\right\rangle \geq \eta \text{ and } \mathbf{p} \in u_j.P \text{ and } \mathbf{p} \in \textsc{Query}(\mathbf{q}, \mathcal{T}_{u_j(\mathbf{p})}, x)\right] \\
&= \mathbb{P}\left[\mathbf{p} \in \textsc{Query}(\mathbf{q}, \mathcal{T}_{u_j(\mathbf{p})}, x) \mid \left\langle u_j.g, \frac{\mathbf{q}-o}{||\mathbf{q}-o||}\right\rangle \geq \eta \text{ and } \mathbf{p} \in u_j.P\right] \\
&\quad \cdot \mathbb{P}\left[\left\langle u_j.g, \frac{\mathbf{q}-o}{||\mathbf{q}-o||}\right\rangle \geq \eta \text{ and } \mathbf{p} \in u_j.P\right] \\
&\geq 0.9 \cdot \mathbb{P}\left[\left\langle u_j.g, \frac{\mathbf{q}-o}{||\mathbf{q}-o||}\right\rangle \geq \eta \text{ and } \mathbf{p} \in u_j.P\right] \\
&\geq 0.9 \cdot G(||\mathbf{q}' - \mathbf{p}.new||/R, \eta) \\
&\geq 0.9 \cdot G\left(\textsc{Project}(x + \delta, R_2, R)/R, \eta\right) \\
&\geq 0.9 \cdot G(x'/R, \eta).
\end{aligned}
$$

The first inequality holds by induction. The second inequality holds by Definition 3.3.1. The third inequality holds since $||\mathbf{q}' - \mathbf{p}.new|| \leq \textsc{Project}(x + \delta, R_2, R)$. Also, the last inequality holds since $\Delta \geq \delta$. Then, we have

$$
\begin{aligned}
&\mathbb{P}\left[\textsc{Query}(\mathbf{q}, \mathcal{T}_{u_j(\mathbf{p})}, x) \text{ will not be called or } \mathbf{p} \notin u_j(\mathbf{p}).P \text{ or } \mathbf{p} \notin \textsc{Query}(\mathbf{q}, \mathcal{T}_{u_j(\mathbf{p})}, x)\right] \\
&\leq 1 - 0.9 \cdot G(x'/R, \eta).
\end{aligned}
$$

Consequently

$$
\begin{aligned}
\mathbb{P}\left[\mathbf{p} \notin \textsc{Query}(\mathbf{q}, \mathcal{T}', x)\right] &\leq (1 - 0.9 \cdot G(x'/R, \eta))^N \\
&\leq 0.1,
\end{aligned}
$$

where the second inequality uses the following fact that since $x'' \geq x'$, we have

$$
N = \left\lceil \frac{100}{G(x''/R, \eta)} \right\rceil \geq \left\lceil \frac{100}{G(x'/R, \eta)} \right\rceil \geq \frac{100}{G(x'/R, \eta)}.
$$

So the inductive step goes through, and the statement of the lemma holds. Now, by taking the union

bound over the failure probability of the Andoni-Indyk round (which succeeds with high probability) and the failure probability of the data dependent part, we succeed by probability at least $1-0.1-0.1=0.8$. □

For points beyond $\delta_x\lfloor\frac{\sqrt{2}}{\delta_x}\rfloor$ we just sample the data set with rate $\frac{1}{n}$ and just store the sampled set (see line 10 in Algorithm 13). In the query procedure we just scan the sub-sampled data set for recovering these points (see line 10 of Algorithm 14). We repeat this procedure $O(\log n)$ times to boost the success probability to high probability. After recovering the sampled points from the various bands using corresponding data structures, Algorithm 14 applies the standard procedure of importance sampling by calculating Z_μ.

Algorithm 13 PREPROCESS-KDE: P is the data-set

1: **procedure** PREPROCESS-KDE(P,μ)
2: $\delta_x \leftarrow 10^{-8}$ ▷ Step size for grid over x
3: $K_1 \leftarrow \lceil\frac{C\log n}{\epsilon^2}\cdot\mu^{-4\delta_x}\rceil$ ▷ where C is some large enough constant
4: **for** $k=1,2,\ldots,K_1$ **do**
5: **for** $j=1,\ldots,\lfloor\sqrt{2}/\delta_x\rfloor$ **do** ▷ Uniform grid with step size δ_x over $[0,\sqrt{2}]$
6: $x \leftarrow j\cdot\delta_x$
7: $\widetilde{P}_{k,x} \leftarrow$ sample each point in P with probability $\min\left\{\frac{1}{n}\left(\frac{1}{\mu}\right)^{1-x^2/2},1\right\}$
8: **for** $i=1,\ldots,10\log n$ **do**
9: $\mathcal{T}_{x,k,i} \leftarrow$ PREPROCESS($\widetilde{P}_{k,x},x,\mu$)
10: $\widetilde{P}_k \leftarrow$ sample each point in P with probability $\frac{1}{n}$
11: Store $\widetilde{P}_k$ ▷ This set will be used to recover points beyond $\delta_x\lfloor\sqrt{2}/\delta_x\rfloor$.

Below, we present the proof of correctness for the outer algorithm, which is very similar to the proof in Section 3.4.

Claim C.3.1 (Unbiasedness of the estimator)**.** *The estimator $Z_{\mu,k}$ for any $\mu\geq\mu^*$ and any $k\in[K_1]$ (see line 6 of Algorithm 14) satisfies the following:*

$$(1-n^{-9})n\mu^* \leq \mathbb{E}[Z_{\mu,k}] \leq n\mu^*.$$

Proof. First note that if a point $\mathbf{p}\in\widetilde{P}_{k,x}$ for some k and x in line 7 of Algorithm 13 is such that $||\mathbf{q}-\mathbf{p}||\in[x-\delta_x,x)$, then since we are preparing $10\log n$ data structures, alongside with Lemma C.3.1 with probability at least $1-n^{-10}$, $\mathbf{p}\in S_x$ (see line 16 of Algorithm 14). Taking union bound over all the points, with probability $1-n^{-9}$, any point in distance $[x-\delta_x,x)$ is being sampled with probability $\min\left\{\frac{1}{n}\left(\frac{1}{\mu}\right)^{1-x^2/2},1\right\}$ for any $x\in\{\delta_x,2\delta_x,\ldots\}\cap(0,\sqrt{2})$. We call this event $\mathscr{E}$. Now, since $p_i\cdot(1-n^{-9})\leq\mathbb{P}[\chi_i=1]\leq p_i$, we have

$$\mathbb{E}[Z_{\mu,k}] = \mathbb{E}\left[\sum_{i=1}^{n}\chi_i\frac{w_i}{p_i}\right] \geq (1-n^{-9})\sum_{i=1}^{n}w_i = (1-n^{-10})n\mu^*.$$

and

$$\mathbb{E}[Z_{\mu,k}] \leq n\mu^*$$

Algorithm 14 QUERY-KDE: **q** is the query point

procedure QUERY-KDE($\mathbf{q}, \mu$)
  $\delta_x \leftarrow 10^{-8}$
  $C_x \leftarrow \lfloor \frac{\sqrt{2}}{\delta_x} \rfloor$
  $K_1 \leftarrow \lceil \frac{C \log n}{\epsilon^2} \cdot \mu^{-4\delta_x} \rceil$ ▷ where C is some large enough constant
  $Z_\mu \leftarrow 0$
  for $k = 1, 2, \ldots, K_1$ **do**
    $Z_{\mu,k}, \leftarrow 0$
    for $j = 1, \ldots, C_x$ **do**
      $x \leftarrow j \cdot \delta_x$
      $S_x \leftarrow \emptyset$
      for $i = 1, \ldots, 10 \log n$ **do**
        $\mathcal{T}_{x,k,i} \leftarrow$ the data structure prepared by line 9 of Algorithm 13
        $P_{x,k,i} \leftarrow$ QUERY($\mathbf{q}, \mathcal{T}_{x,k,i}, x$)
        for $\mathbf{p} \in P_{x,k,i}$ **do**
          if $||\mathbf{q} - \mathbf{p}|| \in [x - \delta_x, x)$ **then**
            $S_x \leftarrow S_x \cup \{\mathbf{p}\}$
      for $\mathbf{p} \in S_x$ **do**
        $\hat{x} \leftarrow ||\mathbf{q} - \mathbf{p}||$
        $Z_{\mu,k} \leftarrow Z_{\mu,k} + \left(\mu^{\frac{\hat{x}^2}{2}}\right) \left(\min\left\{\frac{1}{n} \exp_\mu\left(1 - \frac{x^2}{2}\right), 1\right\}\right)^{-1}$
    for $p \in \widetilde{P}_k$ **do** ▷ Importance sampling for points beyond $\delta_x C_x$.
      if $||\mathbf{q} - \mathbf{p}|| \geq \delta_x C_x$ **then**
        $\hat{x} \leftarrow ||\mathbf{q} - \mathbf{p}||$
        $Z_{\mu,k} \leftarrow Z_{\mu,k} + n\left(\mu^{\hat{x}^2/2}\right)$
    $Z_\mu \leftarrow Z_\mu + \frac{Z_{\mu,k}}{K_1}$
  return Z_μ

where p_i is the probability of sampling i'th point, and χ_i is the indicator for the event that i'th point is recovered. □

We proved that our estimator is unbiased[2] for **any choice** of $\mu \geq \mu^*$. Therefore if $\mu \geq 4\mu^*$, by Markov's inequality the estimator outputs a value larger than μ at most with probability $1/4$. We perform $O(\log n)$ independent estimates, and conclude that μ is higher than μ^* if the median of the estimated values is below μ. This estimate is correct with high probability, which suffices to ensure that we find a value of μ that satisfies $\mu/4 < \mu^* \leq \mu$ with high probability by starting with $\mu = n^{-\Theta(1)}$ (since our analysis assumes $\mu^* = n^{-\Theta(1)}$) and repeatedly halving our estimate (the number of times that we need to halve the estimate is $O(\log n)$ assuming that μ is lower bounded by a polynomial in n, an assumption that we make).

Claim C.3.2. *For μ such that $\mu/4 \leq \mu^* \leq \mu$,* QUERY-KDE($\mathbf{q}, \mu$) *(Algorithm 14) returns a $(1\pm\epsilon)$-approximation to μ^*.*

Proof. Also, one should note that $Z_{\mu,k} < n^2\left(\frac{1}{\mu}\right)$ which implies

$$\mathbb{E}\left[Z_{\mu,k}|\mathscr{E}\right]\cdot\mathbb{P}[\mathscr{E}] + n^2\left(\frac{1}{\mu}\right)(1-\mathbb{P}[\mathscr{E}]) \geq \mathbb{E}[Z_{\mu,k}]$$

So,

$$\mathbb{E}[Z_{\mu,k}]|\mathscr{E}] \geq \left((1-n^{-10})n\mu^* - \frac{1}{n^2}\frac{1}{\mu}\right) = n\mu^* - o(1/n^5)$$

Also, since $Z_{\mu,k}$ is a non-negative random variable, we have

$$\mathbb{E}\left[Z_{\mu,k}|\mathscr{E}\right] \leq \frac{\mathbb{E}\left[Z_{\mu,k}\right]}{\mathbb{P}[\mathscr{E}]} \leq \frac{n\mu^*}{\mathbb{P}[\mathscr{E}]} = n\mu^* + o(1/n^5)$$

Also,

$$\begin{aligned}
\mathbb{E}[Z_{\mu,k}^2] &= \mathbb{E}\left[\left(\sum_{i\in[n]}\chi_i\frac{w_i}{p_i}\right)^2\right] \\
&= \sum_{i\neq j}\mathbb{E}\left[\chi_i\chi_j\frac{w_iw_j}{p_ip_j}\right] + \sum_{i\in[n]}\mathbb{E}\left[\chi_i\frac{w_i^2}{p_i^2}\right] \\
&\leq \sum_{i\neq j} w_iw_j + \sum_{i\in[n]}\frac{w_i^2}{p_i}\mathbb{I}[p_i=1] + \sum_{i\in[n]}\frac{w_i^2}{p_i}\mathbb{I}[p_i\neq 1] \\
&\leq \left(\sum_i w_i\right)^2 + \sum_i w_i^2 + \max_i\left\{\frac{w_i}{p_i}\mathbb{I}[p_i\neq 1]\right\}\sum_{i\in[n]} w_i \\
&\leq 2n^2\mu^{*2} + n^2\left(\frac{1}{\mu}\right)^{-1+4\delta_x}\cdot\mu^* \leq n^2\mu^{2-4\delta_x}
\end{aligned}$$

[2]Up to some small inverse polynomial error.

and

$$\mathbb{E}[Z_{\mu,k}^2|\mathscr{E}] \leq \frac{\mathbb{E}[Z_{\mu,k}^2]}{\mathbb{P}[\mathscr{E}]} \leq n^2\mu^{2-4\delta_x} + o(1/n^5)$$

So in order to get a $(1\pm\epsilon)$-factor approximation to $n\mu$, with high probability, it suffices to repeat the whole process $K_1 = \frac{C\log n}{\epsilon^2}\cdot\mu^{-4\delta_x}$ times (see Algorithm 13 and Algorithm 14), where C is a universal constant. □

C.4 Omitted discussion from Section 3.6.1

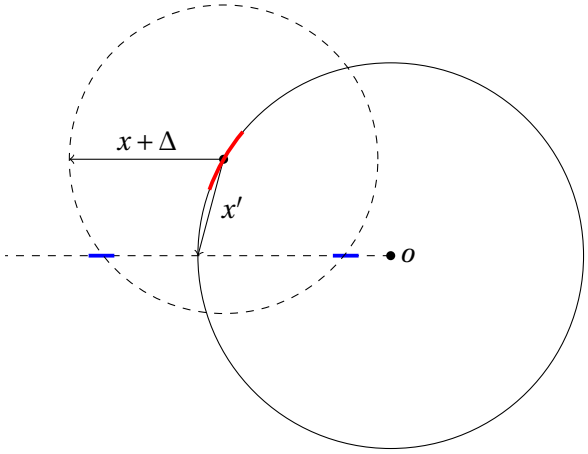


Figure C.3: Geometric illustration of equation $x' = \textsc{Project}(x+\Delta, \widetilde{\ell}, r)$ when we have access to an approximation of x' (red arc).

Given query $\mathbf{q}$, and a LSH node v with $(v.x, v.r) = (x'', r)$, we define $b \in \{1,2\}$ which we use in the definition of the path geometry (Definition 3.6.1). Let $\widetilde{\ell} := ||\mathbf{q} - o||$, where o is the center of the sphere. Note that, if we solve $x' = \textsc{Project}(x+\Delta, \ell, r)$ for ℓ we get the following roots for this equation.

$$\ell_1 = \frac{\sqrt{4r^2((x+\Delta)^2 - x'^2) + x'^4} + 2r^2 - x'^2}{2r} \tag{C.8}$$

and

$$\ell_2 = \frac{-\sqrt{4r^2((x+\Delta)^2 - x'^2) + x'^4} + 2r^2 - x'^2}{2r} \tag{C.9}$$

Stability of ℓ_1 and ℓ_2 for small changes of x': Let $\widetilde{x}'$ be such that $\widetilde{x}' \in [x', x'+\delta']$. Since $\delta' = o(1)$, $r = \Theta(1)$ and $x = \Theta(1)$, if we solve equation $\widetilde{x}' = \textsc{Project}(x+\Delta, \ell, r)$ for ℓ then we get roots $\widetilde{\ell}_1$ and $\widetilde{\ell}_2$ such that $\widetilde{\ell}_1 \in (\ell_1 - \delta'^{1/3}, \ell_1 + \delta^{1/3})$ and $\widetilde{\ell}_2 = (\ell_2 - \delta'^{1/3}, \ell_2 + \delta'^{1/3})$ for large enough n, since $\delta' = \exp(-(\log\log n)^C)$ (see line 10 of Algorithm 8).

Suppose that we solve $x' = \textsc{Project}(x+\Delta, \ell, r)$ for ℓ for all values of $x' \in [x''-\delta', x'']$, and let ℓ_1^* be the

largest quantity that we get by (C.8) and let ℓ_2^* be the largest quantity that we get by (C.9). More formally,

$$\ell_1^* := \max_{x' \in [x''-\delta', x'']} \frac{\sqrt{4r^2((x+\Delta)^2 - x'^2) + x'^4} + 2r^2 - x'^2}{2r}$$

and

$$\ell_2^* := \max_{x' \in [x''-\delta', x'']} \frac{-\sqrt{4r^2((x+\Delta)^2 - x'^2) + x'^4} + 2r^2 - x'^2}{2r}.$$

Now, if $\widetilde{\ell} \in [\ell_1^* - \delta'^{1/3}, \ell_1^*]$ then we let $b = 1$ and $\ell := \ell_1^*$, and otherwise we let $b = 2$ and $\ell := \ell_2^*$. Note that when $b = 2$ it is guaranteed that $\widetilde{\ell} \in [\ell_2^* - \delta'^{1/3}, \ell_2^*]$. One should note that since we define geometry for root to leaf paths, then it is guaranteed that $x' = \textsc{Project}(x + \Delta, \ell, r)$ has at least a real valued solution for ℓ, because otherwise such a root to leaf path is not possible in the tree that the query explores. Also, note that the maximizations above are over the real values, and we ignore the imaginary solutions.

C.5 Omitted claims and proofs from Section 3.6

Claim C.5.1. *Given query* $\mathbf{q}$ *and a pseudo random sphere with geometry* (x'', r, b) *that induces distance* ℓ *let* $\mathbf{q}'$ *be the projection of* $\mathbf{q}$ *on the sphere. In that case, if a point* $\mathbf{p}.new$ *on the sphere is such that* $||\mathbf{q}' - \mathbf{p}.new|| \in (r(\sqrt{2}-\gamma), r(\sqrt{2}+\gamma))$*, then*

$$||\mathbf{p} - \mathbf{q}|| \in \left(c - r\psi, c + r\psi\right)$$

where $\psi := \gamma^{1/3} + \delta'^{1/4} + \delta^{1/4}$, $c := \sqrt{\ell^2 + r^2}$.

Proof. Since $\mathbf{q}$ and the geometry of the sphere induce distance ℓ, then $||\mathbf{q} - o|| \in [\ell - \delta'^{1/3}, \ell]$. Now, suppose that we move $\mathbf{q}$ in the direction of the vector from o to $\mathbf{q}$ and reach a point $\widetilde{\mathbf{q}}$ such that $||\widetilde{\mathbf{q}} - o|| = \ell$. Then, by assumption

$$\textsc{Project}(||\widetilde{\mathbf{q}} - \mathbf{p}.new||, \ell, r) \in (r(\sqrt{2}-\gamma), r(\sqrt{2}+\gamma)).$$

Let $\widetilde{y} := ||\widetilde{\mathbf{q}} - \mathbf{p}.new||$. Then,

$$\frac{r}{\ell}\left(\widetilde{y}^2 - (\ell - r)^2\right) \in \left(r^2(\sqrt{2}-\gamma)^2, r^2(\sqrt{2}+\gamma)^2\right)$$

which using the definition $c := \sqrt{r^2 + \ell^2}$ (see Figure 3.10) translates to

$$\begin{aligned}\widetilde{y}^2 &\in \left(r\ell(2 - 2\sqrt{2}\gamma + \gamma^2) + (\ell - r)^2, r\ell(2 + 2\sqrt{2}\gamma + \gamma^2) + (\ell - r)^2\right) \\ &= \left(c^2 - 2\sqrt{2}r\ell\gamma + r\ell\gamma^2, c^2 + 2\sqrt{2}r\ell\gamma + r\ell\gamma^2\right)\end{aligned}$$

which also translates to

$$\widetilde{y} \in \left(\sqrt{c^2 - 2\sqrt{2} r\ell\gamma + r\ell\gamma^2}, \sqrt{c^2 + 2\sqrt{2} r\ell\gamma + r\ell\gamma^2} \right)$$

Now, noting that $\ell = O(1)$ and $r = \Theta(1)$, for large enough n we get that

$$\begin{aligned} \sqrt{c^2 - 2\sqrt{2} r\ell\gamma + r\ell\gamma^2} &\geq c - \sqrt{2\sqrt{2} r\ell\gamma - r\ell\gamma^2} \\ &\geq c - r\gamma^{1/3}. \end{aligned}$$

And Similarly,

$$\begin{aligned} \sqrt{c^2 + 2\sqrt{2} r\ell\gamma + r\ell\gamma^2} &\leq c - \sqrt{2\sqrt{2} r\ell\gamma + \ell\gamma^2} \\ &\leq c + r\gamma^{1/3}. \end{aligned}$$

So, overall

$$\widetilde{y} \in \left(c - r\gamma^{1/3}, c + r\gamma^{1/3} \right)$$

Noting that $||\mathbf{q} - \widetilde{\mathbf{q}}|| \leq \delta'^{1/3}$ and $||\mathbf{p} - \mathbf{p}.new|| \leq \delta$, using the triangle inequality, for $y := ||\mathbf{q} - \mathbf{p}||$ we get

$$y \in \left(c - r\gamma^{1/3} - \delta'^{1/3} - \delta, c + r\gamma^{1/3} + \delta'^{1/3} + \delta \right)$$

Again noting that $r = \Theta(1)$ and setting $\psi = \gamma^{1/3} + \delta'^{1/4} + \delta^{1/4}$

$$y \in \left(c - r\psi, c + r\psi \right).$$

Note that in this proof we did not optimize the inequalities and we were generous in bounding variables for the sake of brevity. □

Claim C.5.2. *Let y be such that $y \geq x + \Delta$ for some $x \in (\delta_x, \sqrt{2})$, and y'' is such that*

$$y'' \in \left[\textsc{Project}(y - \delta, R_2, R), \textsc{Project}(y + \delta, R_2, R) \right]$$

for some R_2 and R. Let $x' = \textsc{Project}(x + \Delta, R_2, R)$, and let x'' be the smallest element in W_x which is not larger than x'. Additionally, assume that we have the following properties:

(p1) $\frac{\delta}{\Delta} = o(1)$

(p2) $\frac{\delta'}{\Delta} = o(1)$

(p3) $\Delta = \Theta(1)$

(p4) $x' \leq \frac{8}{5} \cdot R$

If η is such that $\frac{F(\eta)}{G(x''/R,\eta)} = \exp_\mu\left(\frac{1}{T}\right)$, then, we have **(a)**

$$\frac{G(y''/R,\eta)}{F(\eta)} \le \exp_\mu\left(-(1-o(1))\frac{4(R/x')^2-1}{4(R/y')^2-1}\cdot\frac{1}{T}\right)$$

and Furthermore, **(b)** *when $R = O(1)$, then*

$$\frac{G(y''/R,\eta)}{F(\eta)} \le \exp_\mu\left(-\frac{4(R/x')^2-1}{4(R/y')^2-1}\cdot\frac{1}{T}\right).$$

Proof. By assumption we have

$$\frac{F(\eta)}{G(x''/R,\eta)} = \exp_\mu\left(\frac{1}{T}\right). \tag{C.10}$$

On the other hand, if we set $s = x''/R$

$$\begin{aligned}\frac{F(\eta)}{G(x''/R,\eta)} &= \frac{e^{-(1+o(1))\frac{\eta^2}{2}}}{e^{-(1+o(1))\frac{\eta^2}{2}\cdot\frac{4}{4-s^2}}}\\ &= e^{(1+o(1))\frac{\eta^2}{2}\cdot\frac{s^2}{4-s^2}}\\ &= \exp\left((1+o(1))\frac{\eta^2}{2}\cdot\frac{x''^2}{4R^2-x''^2}\right).\end{aligned} \tag{C.11}$$

By triangle inequality in Euclidean space (see Figure C.4) we have,

$$\begin{aligned}x' &\ge (x+\Delta) - |R_2 - R| && (C.12)\\ &\ge (x+\Delta) - (x+\delta) && \text{By line 20 of Algorithm 10}\\ &= \Delta - \delta = (1-o(1))\Delta && \text{By property } \textbf{(p1)}.\end{aligned}$$

Also note that by assumption

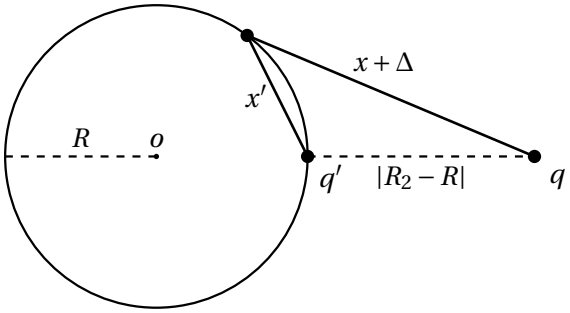
$$x'' \in \left[x', x'+\delta'\right],$$

then, since $\frac{\delta'}{\Delta} = o(1)$ by property **(p2)**, we have

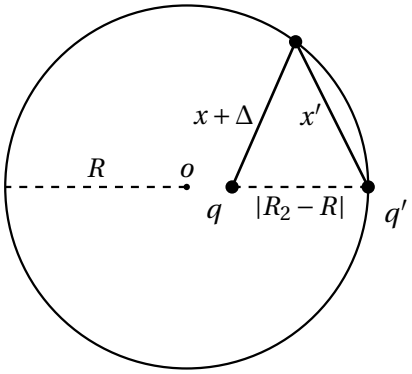
$$x'' = (1 \pm o(1)) \cdot x'.$$

And by property **(p4)**, we have

$$\frac{x''^2}{4R^2-x''^2} = (1 \pm o(1)) \cdot \frac{x'^2}{4R^2-x'^2}$$



(a) When the query is outside the sphere



(b) When the query is inside the sphere

Figure C.4: Triangle inequality instances for (C.12)

Therefore

$$\frac{F(\eta)}{G(x'/R,\eta)} = \exp\left((1+o(1))\frac{\eta^2}{2}\cdot\frac{x'^2}{4R^2-x'^2}\right) \tag{C.13}$$

$$= \exp_\mu\left((1\pm o(1))\frac{1}{T}\right). \tag{C.14}$$

On the other hand, if we set $s' = y'/R$, similarly

$$\begin{aligned}\frac{F(\eta)}{G(y'/R,\eta)} &= \frac{e^{-(1+o(1))\frac{\eta^2}{2}}}{e^{-(1+o(1))\frac{\eta^2}{2}\cdot\frac{4}{4-s'^2}}}\\ &= e^{(1+o(1))\frac{\eta^2}{2}\cdot\frac{s'^2}{4-s'^2}}\\ &= \exp\left((1+o(1))\frac{\eta^2}{2}\cdot\frac{y'^2}{4R^2-y'^2}\right).\end{aligned} \tag{C.15}$$

Thus, by (C.13), (C.14) and (C.15)

$$\begin{aligned}\frac{G(y'/R,\eta)}{F(\eta)} &= \exp_\mu\left(-(1\pm o(1))\frac{1}{T}\frac{4-s^2}{s^2}\cdot\frac{s'^2}{4-s'^2}\right)\\ &= \exp_\mu\left(-(1\pm o(1))\frac{1}{T}\frac{4(R/x')^2-1}{4(R/y')^2-1}\right).\end{aligned} \tag{C.16}$$

Note that

$$y'' \in \left[\textsc{Project}(y-\delta, R_2, R), \textsc{Project}(y+\delta, R_2, R)\right]$$

Then since $\frac{\delta'}{\Delta} = o(1)$ by property **(p2)**, we have

$$y'' \geq \textsc{Project}(y-\Delta/2, R_2, R) = y',$$

Now since $G(s,\eta)$ is monotone decreasing in s, we have

$$\frac{G(y'/R,\eta)}{F(\eta)} \geq \frac{G(y''/R,\eta)}{F(\eta)}. \tag{C.17}$$

Now, by (C.16) and (C.17), we have the statement of the first part of the claim.

For the case when $R = O(1)$, and consequently $R_2 = O(1)$ (by the assumption fact that $x \leq \sqrt{2}$), by property **(p3)**:

$$\begin{aligned} y''^2 - y'^2 &\geq \left(\textsc{Project}\left(y-\delta, R_2, R\right)\right)^2 - \left(\textsc{Project}\left(y-\Delta/2, R_2, R\right)\right)^2 \\ &= \frac{R}{R_2}\left(\left(y-\delta\right)^2 - (R_2-R)^2\right) - \frac{R}{R_2}\left(\left(y-\Delta/2\right)^2 - (R_2-R)^2\right) \\ &= \Omega(1) \end{aligned} \tag{C.18}$$

Then,

$$\begin{aligned} (1-o(1))\cdot\frac{1}{4(R/y'')^2-1} &\geq (1-o(1))\cdot\frac{y'^2}{4R^2-y''^2} && \text{Since } y'' \geq y' \\ &\geq \frac{y'^2}{4R^2-y'^2} && \text{By (C.18)} \\ &\geq \frac{1}{4(R/y')^2-1} \end{aligned}$$

which implies,

$$\exp_\mu\left(-\frac{1}{T}\frac{4(R/x')^2-1}{4(R/y')^2-1}\right) \geq \exp_\mu\left(-(1\pm o(1))\frac{1}{T}\frac{4(R/x')^2-1}{4(R/y'')^2-1}\right) = \frac{G(y''/R,\eta)}{F(\eta)},$$

which proves the second part of the claim. □

Claim C.5.3. *Let V denote the output of* PseudoRandomify(v,γ) *on a node v of a recursion tree $\mathcal{T}$ associated with a dataset P of diameter bounded by D. Then for every positive integer j, where R_{min} is the parameter from line 3 of Algorithm 9, the number of sets with diameter at least $(1-\gamma^2/2)^j D$ contained in V is upper bounded by Λ^j for $\Lambda = O(D\log|P|)/\delta$.*

Proof. Note that an input dataset is first partitioned into at most $\lceil R/\delta \rceil = O(D/\delta)$ spherical shells. For each spherical shell one repeatedly removes dense clusters (containing at least a $1/10$ fraction of the current dataset), repeating this process $O(\log|P|)$ times, since at most 10 clusters are removed before the

dataset size decreases by a constant factor. Every such ball has radius smaller than the original dataset by a $(1-\gamma^2/2)$ factor [13]. This gives the claimed bound. □

We now give

Proof of Lemma 3.6.4: The proof is by induction on (a,b), where a is the number ℓ of LSH nodes on the path from $v \in \mathcal{T}$ to the closest leaf, b is the number of pseudorandomification nodes on such a path and $r = v.R$ is the radius of the dataset. We prove that the expected number of nodes in the subtree of such a node v in $\mathcal{T}$ is upper bounded by

$$(L\cdot\Lambda)^a \cdot (100/\mu^{1/T})^b \cdot \Lambda^j.$$

Here $\Lambda = (O(D\log|P|)/\delta)$ is the parameter from Claim C.5.3, $j = \log_{\frac{1}{1-\gamma^2/2}}(R_{max}/r)$ is an upper bound on the number of times the radius of the sphere could have shrunk through calls to PSEUDORANDOMIFY from the largest possible (bounded R_{max}) to its current value r, and $L = \log_{\frac{1}{1-\gamma^2/2}}(R_{max}/R_{min})$ is the maximum number of times a point can be part of a dataset that PSEUDORANDOMIFY is called on (since the radius reduces by a factor of $1-\gamma^2/2$ in every such call).

The **base** is provided by the case of v being a leaf. We now give the **inductive step**. First suppose that $u \in \mathcal{T}$ is a pseudorandomification node. Let x' denote the value of rounded projected distance computed in line 19 of Algorithm 10. Then Algorithm 8 generates $\frac{100}{G(x'/R,\eta)}$ Gaussians, and the expected number of Gaussians for which the condition in line 29 is satisfied (i.e. the number of children of u that the query q explores) is exactly $\frac{100F(\eta)}{G(x'/R,\eta)}$ by definition of $F(\eta)$ (see Lemma 3.3.1 in Section 3.3). We also have $\frac{F(\eta)}{G(x'/R,\eta)} = (1/\mu)^{1/T}$ by setting of parameters in line 16 of Algorithm 10. Putting this together with the inductive hypothesis and noting that LSH nodes do not change the radius of the sphere, we get that the expected number of nodes of $\mathcal{T}$ that the query explores is bounded by

$$100\left(1/\mu\right)^{1/T} \cdot (L\cdot\Lambda)^a \cdot (100/\mu^{1/T})^{b-1} \cdot \Lambda^j = (L\cdot\Lambda)^a \cdot (100/\mu^{1/T})^b \cdot \Lambda^j,$$

as required.

Now suppose that $u \in \mathcal{T}$ is a pseudorandomification node. Then by Claim C.5.3 for every i the number of datasets with diameter at least $(1-\gamma^2/2)^i r$ generated by PSEUDORANDOMIFY is bounded by Λ^i. For every $i = 0,\ldots,L$ the number of nodes with radius in $((1-\gamma^2/2)^{i-1}r, (1-\gamma^2/2)^i r]$ that are generated is bounded by Λ^{i-1}. For such nodes we have by the inductive hypothesis that the expected number of nodes of $\mathcal{T}$ explored in their subtree is upper bounded by

$$(L\cdot\Lambda)^{a-1}\left(100/\mu^{1/T}\right)^b \cdot \Lambda^{j-i+1}.$$

Summing over all i between 1 and $\log_{\frac{1}{1-\gamma^2/2}}(r/R_{min})$, we get that the total number of nodes that the query

is expected to explore in the subtree of u is bounded by

$$\begin{aligned}\sum_{i=1}^{\log_{\frac{1}{1-\gamma^2/2}}(r/R_{min})} (L\cdot\Lambda)^{a-1}\left(100/\mu^{1/T}\right)^b\cdot\Lambda^{j-i+1}\cdot\Lambda^i &\le L\cdot(L\cdot\Lambda)^{a-1}\left(100/\mu^{1/T}\right)^b\cdot\Lambda^{j+1}\\ &\le (L\cdot\Lambda)\cdot(L\cdot\Lambda)^{a-1}\left(100/\mu^{1/T}\right)^b\cdot\Lambda^{j+1}\\ &\le (L\cdot\Lambda)^a\cdot\left(100/\mu^{1/T}\right)^b\cdot\Lambda^j\end{aligned}$$

proving the inductive step.

Substituting $\alpha^*\cdot T$ as the upper bound on the number of levels in $\mathcal{T}$ as per Algorithm 8, we thus get that the number of nodes explored by the query is bounded by

$$(L\cdot\Lambda)^T\cdot(100/\mu^{1/T})^{\alpha^* T}\cdot\Lambda^L \le (100L\cdot\Lambda)^T\cdot\Lambda^L\cdot(1/\mu)^{\alpha^*} = n^{o(1)}\cdot(1/\mu)^{\alpha^*}$$

in expectation. In the last transition we used the fact that

$$(100L\cdot\Lambda)^T\cdot\Lambda^L = (100\cdot\log_{\frac{1}{1-\gamma^2/2}}(R_{max}/R_{min})\cdot(O(R_{max}\log|P|)/\delta))^{\sqrt{\log n}}\cdot((O(D\log|P|)/\delta)^{\sqrt{\log n}} = n^{o(1)}$$

by our setting of parameters since $\gamma = 1/\log\log\log n$, $R_{max} = O(1)$, $R_{min} = \Omega(1)$ and $\delta = \exp(-(\log\log n)^{O(1)})$ as per Algorithm 8 and Algorithm 9. And also since we use $100\left(\frac{1}{\mu}\right)^\alpha$ Andoni-Indyk hash functions (see Algorithm 7), we get

$$n^{o(1)}\cdot\left(\frac{1}{\mu}\right)^{\alpha^*+\alpha}$$

in total. □

Proof of Claim 3.6.6:

By Lemma 3.3.1 and Lemma 3.3.2 and Definition 3.3.1 one has

$$F(\eta) = e^{-(1+o(1))\cdot\frac{\eta^2}{2}}$$

and

$$G(x'/R,\eta) = e^{-(1+o(1))\cdot\frac{2\eta^2(1-\alpha(x'/R))}{2\beta^2(x'/R)}} = e^{-(1+o(1))\cdot\frac{2\eta^2}{2(1+\alpha(x'/R))}},$$

where $\alpha(x'/R) := 1-\frac{(x'/R)^2}{2}$. Using the assumption that $x' > \Delta$ we get that

$$G(x'/R,\eta) \le e^{-(1+o(1))\cdot\frac{\eta^2}{2-\left((\Delta/R)^2/2\right)}}.$$

And in particular using the fact that $R \ge \Delta$

$$\frac{F(\eta)}{G(x'/R,\eta)} \ge e^{-(1+o(1))\cdot\frac{\eta^2}{2}+(1+o(1))\cdot\frac{\eta^2}{2-\left((\Delta/R)^2/2\right)}} = (G(x'/R,\eta))^{\Omega(\Delta^2)},$$

or, equivalently, $\frac{1}{G(x'/R,\eta)} = \left(\frac{F(\eta)}{G(x'/R,\eta)}\right)^{O(1/\Delta^2)}$. □

Proof of Claim 3.6.5: For $j^* = k_J + 1$, by definition of $f_{z_j,J+1}$ for $i \in \{j^*-1,\dots,I\}$ and the fact that

$$b'_{y,J+1} = \widetilde{B}_{y,J+1} = \widetilde{A}_{y,J},$$

we have

$$f_{z_i,J+1} = \log_{1/\mu}\left(\sum_{y\in D\cap[z_{i+1},z_{i-1})} \widetilde{A}_{y,J}\right) \tag{C.19}$$

On the other hand, (3.40) and the fact that $\widetilde{B}_{y,j} = \mathbb{E}[B_{y,j}]$ we have

$$\sum_{y\geq c_J+\psi R_J} \widetilde{B}_{y,J} \leq \frac{\tau}{1-2\tau}\cdot \sum_{y\in(c_J-\psi R_J,c_J+\psi R_J)} \widetilde{B}_{y,J}.$$

Also recall (3.41), where we have

$$\widetilde{A}_{y,J} = \widetilde{B}_{y,J}\cdot p_{y,J}$$

where $p_{y,J}$ is a decreasing and non-negative function in y (for the valid range of y). This implies that

$$\sum_{y\geq c_J+\psi R_J} \widetilde{A}_{y,J} \leq \frac{\tau}{1-2\tau}\cdot \sum_{y\in(c_J-\psi R_J,c_J+\psi R_J)} \widetilde{A}_{y,J}.$$

Now, we have

$$\begin{aligned}
\sum_y \widetilde{A}_{y,J} &= \sum_{y<c_J+\psi R_J} \widetilde{A}_{y,J} + \sum_{y\geq c_J+\psi R_J} \widetilde{A}_{y,J} \\
&\leq \sum_{y<c_J+\psi R_J} \widetilde{A}_{y,J} + \frac{\tau}{1-2\tau}\cdot \sum_{y\in(c_J-\psi R_J,c_J+\psi R_J)} \widetilde{A}_{y,J} \\
&\leq \sum_{y\leq z_{j^*-1}} \exp_\mu\left(f_{y,J+1}\right) + \exp_\mu\left(f_{z_{j^*-1},J+1}\right) \\
&\leq\ O(1)\cdot\exp_\mu(7\delta_z) = \exp_\mu(7\delta_z + o(1))
\end{aligned}$$

where the second inequality is based on Definition 3.6.7, (C.19) and setting of parameters (the fact that $\psi = o(1)$, $\delta_z = \Theta(1)$ and $R_{\max} = O(1)$). The last inequality is by the assumption that $f_{y,J+1} < 7\delta_z$ for $y \leq z_{j^*-1}$. □

C.6 Proof of Claim 3.8.3

Proof of Claim 3.8.3: We want to prove that

$$\left(\frac{4\left(\frac{r_j}{x'}\right)^2-1}{4\left(\frac{r_j}{y'}\right)^2-1}\right)\left(\frac{2\left(\frac{z_{k_j}}{z_i}\right)^2-1}{2\left(\frac{z_{k_j}}{x}\right)^2-1}\right) \geq (1-10^{-4}).$$

By defining $z := z_{k_j}$, $s := z_i$ and $r := r_j$ for the sake of brevity, the left hand side becomes

$$\left(\frac{x^2}{x'^2}\right)\cdot\left(\frac{y'^2}{s^2}\right)\cdot\left(\frac{4r^2-x'^2}{2z^2-x^2}\right)\left(\frac{2z^2-s^2}{4r^2-y'^2}\right).$$

We upper-bound each term one by one.

First term: Since $x' := x+\Delta$ then

$$\frac{x}{x'} = \frac{x}{x+\Delta} = 1-\frac{\Delta}{x+\Delta} \geq 1-\frac{\Delta}{\delta_x+\Delta} \geq 1-\frac{\Delta}{\delta_x} = 1-10^{-12}$$

where we used the fact that $x \geq \delta_x$, and the last transition is by the setting of parameters. So,

$$\frac{x^2}{x'^2} \geq 1-10^{-11}$$

Second term: Since $y' := y-\Delta/2$ and $y \in \left(s(1+\delta_z)^{-1}, s(1+\delta_z)\right)$ then

$$\frac{y'}{s} \geq \frac{y-\Delta/2}{y(1+\delta_z)} \geq \frac{1-\delta_z}{1+\delta_z} \geq 1-3\delta_z$$

where we used the fact that $y \geq \delta_x$ (since $y \geq x$) and also considered that by the parameter setting $\delta_z = 10^{-6}$, $\delta_x = 10^{-8}$ and $\Delta = 10^{-20}$. Consequently, we have

$$\frac{y'^2}{s^2} \geq 1+9\delta_z^2-6\delta_z \geq 1-10^{-5}.$$

Third term: Note that by (3.31) we have

$$r(\sqrt{2}+\psi) \in [z, z(1+\delta_z))$$

which combining with the fact that $\psi = o(1)$ implies

$$r \in \left[\frac{z(1-o(1))}{\sqrt{2}}, \frac{z(1+\delta_z)}{\sqrt{2}}\right). \tag{C.20}$$

Note that we used the fact that $z \geq x$ so $z = \Omega(1)$ (actually we have $z = \Theta(1)$). On the other hand, by the bound for the **first term** we have

$$x'^2 \leq x^2\left(1+10^{-10}\right)$$

Now, we use these tools to bound the third term[3]:

$$\begin{aligned}\frac{4r^2-x'^2}{2z^2-x^2} &\geq \frac{2z^2(1-o(1))-x^2(1+10^{-10})}{2z^2-x^2}\\ &\geq 1-\frac{2o(1)z^2+10^{-10}x^2}{2z^2-x^2}\\ &\geq 1-\frac{2\times 10^{-10}x^2}{2z^2-x^2}\\ &\geq 1-10^{-9}\end{aligned}$$

Fourth term: For the fourth term, actually its easier to upper-bound the inverse of it. First, note that

$$y' = y-\Delta/2 \geq y(1-\delta_z) \geq s(1-\delta_z)(1+\delta_z)^{-1} \geq s(1-10^{-5}).$$

The first inequality is due to $y \geq x \geq \delta_x = 10^{-8}$ and $\delta_z = 10^{-6}$. This also implies that

$$y'^2 \geq s^2(1-3\times 10^{-5}).$$

On the other hand, by (C.20) we have

$$2r^2 \leq z^2(1+\delta_z)^2 \leq z^2(1+10^{-5}).$$

Combining these facts we have

$$\begin{aligned}\frac{4r^2-y'^2}{2z^2-s^2} &\leq \frac{2z^2(1+10^{-5})-s^2(1-3\times 10^{-5})}{2z^2-s^2}\\ &\leq 1+10^{-5}+4\times 10^{-5}\frac{s^2}{2z^2-s^2}\\ &\leq 1+5\times 10^{-5}\end{aligned}$$

where the last transition is due to the fact that $s \leq z$ (or $z_i \leq z_{k_j}$ equivalently). Therefore, we have a lower-bound of $1-5\times 10^{-5}$ for the fourth term.

Combining the bounds: Now, we have:

$$\begin{aligned}\left(\frac{x^2}{x'^2}\right)\cdot\left(\frac{y'^2}{s^2}\right)\cdot\left(\frac{4r^2-x'^2}{2z^2-x^2}\right)\left(\frac{2z^2-s^2}{4r^2-y'^2}\right) &\geq (1-10^{-11})(1-10^{-5})(1-10^{-9})(1-5\times 10^{-5})\\ &\geq 1-10^{-4}\end{aligned}$$

which proves the claim. □

[3]Note that for the sake of brevity we are being generous in bounding terms and the inequalities are not tight

General Kernels

Lemma C.6.1 (Uniqueness of Maximum)**.** *Let $f:[a,b]\to\mathbb{R}$ be a three times differentiable function in (a,b) such that:*

- $f(a)<0$
- $\exists y'\in(a,b]$ *such that* $f(y')>0$
- *for all* $y\in(a,b)$ *it holds* $\frac{d^3}{dy^3}f(y)\leq 0$.

Then

1. $\exists y^*\in(a,y')$ *such that* $f(y^*)=0$.
2. $\exists\eta\in(y^*,b]$ *such that* η *is the unique maximum of* f *in* $[a,b]$ *and the function is monotone increasing in* $[a,\eta]$.

Proof. We prove the statements in order:

1. Using the first two assumptions and continuity of f (since it is differentiable) we get by the Intermediate Value Theorem that $\exists y^*\in(a,y')$ such that $f(y^*)=0$.
2. Since the function is defined on a closed interval it attains a maximum. We show that there exists only one maximum. Assume that there exist two local maxima $\eta_1<\eta_2\in(a,b]$. Then, there must be a local minimum $\eta_0\in(\eta_1,\eta_2)$ for which $f''(\eta_0)>0$. However, this is impossible since $f''(\eta_1)<0$ and the function f'' is non-increasing. Hence, there is exactly one local maximum η in $(a,b]$ and the function is increasing in $[a,\eta]$ (and decreasing in $(\eta,b]$ if $\eta\neq b$).

□

Corollary C.6.1. *Let $\phi:\mathbb{R}_+\to\mathbb{R}$ be any function such that $\phi'''(y)\leq 0$. For all $x>0$, $T\geq 1$ and $c_1\geq c_2\geq\ldots\geq c_t>\frac{x}{\sqrt{2}}$ such that $\exists y'\in(x,\sqrt{2}c_t]$ with $f(y')>0$ define:*

$$f(y):=\big[\phi(y)-\phi(x)\big]-\sum_{s=1}^{t}\frac{2(c_s/x)^2-1}{2(c_s/y)^2-1}\cdot\frac{1}{T}.$$

Then, the conclusion of Lemma C.6.1 holds. In particular, it holds for all $\phi(y)\propto(y)^p$ with $p\leq 2$.

Proof. Follows by observing that the second derivative of the summation term is decreasing and that $\phi'''(y)\propto-(2-p)p\cdot(p-1)\frac{1}{y^{2-p}}\leq 0$ for all $p\leq 2$ and $y>0$. □

Claim C.6.1 (Monotonicity)**.** *For every $i\in[|R|]$ and $\phi:\mathbb{R}_+\to\mathbb{R}$ as in Corollary C.6.1 we have*

(a) *there exists a* $y^* \in (x, \sqrt{2})$ *such that* $g_{y^*,j_i} \geq 0$, $g_{y,j_i} \leq 0$ *for any* $y \in Z_x$ *such that* $y \leq y^*$, *and* g_{y,j_i} *is non-decreasing in* y *for* $y \in [y^*, z_{j_i}]$;

(b) *there exists a* $y^* \in (x, \sqrt{2})$ *such that* $h^{(N-1)}_{y^*} \geq 0$, $h^{(N-1)}_{y} \leq 0$ *for any* $y \in Z_x$ *such that* $y \leq y^*$ *and* $h^{(N-1)}_{y}$ *is non-decreasing in* y *for* $y \in [y^*, z_{j_i}]$.

Proof. Let

$$q(y) := \sum_{i=1}^{t} \frac{2(c_s/x)^2 - 1}{2(c_s/y)^2 - 1} \frac{1}{T},$$

where $c_1 \geq c_2 \geq \ldots \geq c_t \geq z_1 \geq x$ for some $z_1 \geq x$. And let y_1^* be such that $\phi(y_1^*) - \phi(x) - q(y_1^*) = 0$ and let $\widetilde{y}_1$ be the smallest value such that $\widetilde{y}_1 \geq y_1^*$ and $\phi(\widetilde{y}_1) - \phi(x) - q(\widetilde{y}_1) = \theta$ for some $\theta \geq 0$. Now define $G_1(y)$ on $[y_1^*, z_1]$, for some $z_1 \geq \widetilde{y}_1$ as follows

$$G_1(y) := \begin{cases} \phi(y) - \phi(x) - q(y) & y \in [y_1^*, \widetilde{y}_1) \\ \theta & y \in [\widetilde{y}_1, z_1] \end{cases} \tag{C.21}$$

See the red curve in Figure 3.11.

Also, let $\hat{q}(y) := \frac{2(z_1/x)^2-1}{2(z_1/y)^2-1} \frac{1}{T}$. Let $y_2^* \geq y_1^*$ such that $G_1(y_2^*) - \hat{q}(y_2^*) = 0$. Now, we define $G_2(y)$ for $y \in [y_2^*, z_2]$ as follows:

$$G_2(y) := \min\left\{G_1(y) - \hat{q}(y), \theta'\right\}$$

where $\theta' := G_1(z_2) - \hat{q}(z_2)$ and $\theta' \geq 0$ for some $z_2 \leq z_1$. By the definition of y_2^*, function $G_2(y)$ for $y \in [y_2^*, \widetilde{y}_1]$ is in the form of the function in Claim 3.8.4 and thus, it has a unique maximum at some $\eta \in [y_2^*, \widetilde{y}_1]$. Also, recall that $G_1(y) = \theta$ for $y \in [\widetilde{y}_1, z_2]$. Also, one should note that since $\hat{q}(y)$ is a monotone increasing function for $y \in (0, \sqrt{2}z_1)$ and hence for $y \in [\widetilde{y}_1, z_2]$, then $\theta' \leq G_2(\widetilde{y}_1)$ and therefore $\theta' \leq G_2(\eta)$. This guarantees that there exist a $\widetilde{y}_2 \in [y_2^*, \eta]$ such that $G_2(\widetilde{y}_2) = \theta'$. The reason is that $G_2(y)$ is a continuous increasing function for $y \in [\widetilde{y}_2, \eta]$. So, we have

$$G_2(y) := \begin{cases} \phi(y) - \phi(x) - q'(y) & y \in [y_2^*, \widetilde{y}_2) \\ \theta' & y \in [\widetilde{y}_2, z_2] \end{cases} \tag{C.22}$$

where, $q'(y) := q(y) - \hat{q}(y)$. See the blue curve in Figure 3.11.

□

Bibliography

[1] K. J. Ahn, S. Guha, and A. McGregor, "Analyzing graph structure via linear measurements," in Rabani [130], pp. 459–467.

[2] D. R. Karger, "Random sampling in cut, flow, and network design problems," in *Proceedings of the Twenty-Sixth Annual ACM Symposium on Theory of Computing, 23-25 May 1994, Montréal, Québec, Canada* (F. T. Leighton and M. T. Goodrich, eds.), pp. 648–657, ACM, 1994.

[3] A. A. Benczúr and D. R. Karger, "Approximating S-T minimum cuts in $\tilde{O}(n^2)$ time," in *38th Annual Symposium on Theory of Computing*, pp. 47–55, 1996.

[4] D. A. Spielman and S. Teng, "Nearly-linear time algorithms for graph partitioning, graph sparsification, and solving linear systems," in *Proceedings of the 36th Annual ACM Symposium on Theory of Computing, Chicago, IL, USA, June 13-16, 2004*, pp. 81–90, 2004.

[5] J. D. Batson, D. A. Spielman, N. Srivastava, and S. Teng, "Spectral sparsification of graphs: theory and algorithms," *Commun. ACM*, vol. 56, no. 8, pp. 87–94, 2013.

[6] K. J. Ahn, S. Guha, and A. McGregor, "Spectral sparsification in dynamic graph streams," in *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, pp. 1–10, Springer, 2013.

[7] M. Kapralov, Y. T. Lee, C. Musco, C. Musco, and A. Sidford, "Single pass spectral sparsification in dynamic streams," in *55th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2014, Philadelphia, PA, USA, October 18-21, 2014*, pp. 561–570, IEEE Computer Society, 2014.

[8] M. Charikar and P. Siminelakis, "Hashing-based-estimators for kernel density in high dimensions," in *2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS)*, pp. 1032–1043, IEEE, 2017.

[9] M. Charikar and P. Siminelakis, "Multi-resolution hashing for fast pairwise summations," in *2019 IEEE 60th Annual Symposium on Foundations of Computer Science (FOCS)*, IEEE, 2019.

[10] A. Andoni and P. Indyk, "Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions," in *47th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2006), 21-24 October 2006, Berkeley, California, USA, Proceedings* [131], pp. 459–468.

[11] A. Andoni, P. Indyk, H. L. Nguyen, and I. P. Razenshteyn, "Beyond locality-sensitive hashing," in *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2014, Portland, Oregon, USA, January 5-7, 2014* (C. Chekuri, ed.), pp. 1018–1028, SIAM, 2014.

[12] A. Andoni and I. P. Razenshteyn, "Optimal data-dependent hashing for approximate near neighbors," in *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing, STOC 2015, Portland, OR, USA, June 14-17, 2015* (R. A. Servedio and R. Rubinfeld, eds.), pp. 793–801, ACM, 2015.

[13] A. Andoni, T. Laarhoven, I. P. Razenshteyn, and E. Waingarten, "Optimal hashing-based time-space trade-offs for approximate near neighbors," in Klein [132], pp. 47–66.

[14] M. Kapralov, A. Mousavifar, C. Musco, C. Musco, N. Nouri, A. Sidford, and J. Tardos, "Fast and space efficient spectral sparsification in dynamic streams," in *Proceedings of the 2020 ACM-SIAM Symposium on Discrete Algorithms, SODA 2020, Salt Lake City, UT, USA, January 5-8, 2020* (S. Chawla, ed.), pp. 1814–1833, SIAM, 2020.

[15] J. Nelson and H. Yu, "Optimal lower bounds for distributed and streaming spanning forest computation," *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019, San Diego, California, USA, January 6-9, 2019*, pp. 1844–1860, 2019.

[16] A. McGregor, "Graph sketching and streaming: New approaches for analyzing massive graphs," in *Computer Science - Theory and Applications - 12th International Computer Science Symposium in Russia, CSR 2017, Kazan, Russia, June 8-12, 2017, Proceedings* (P. Weil, ed.), vol. 10304 of *Lecture Notes in Computer Science*, pp. 20–24, Springer, 2017.

[17] Y. Li, H. L. Nguyen, and D. P. Woodruff, "Turnstile streaming algorithms might as well be linear sketches," in *Proceedings of the 46th Annual ACM Symposium on Theory of Computing (STOC)*, pp. 174–183, 2014.

[18] K. J. Ahn, S. Guha, and A. McGregor, "Graph sketches: sparsification, spanners, and subgraphs," in Benedikt *et al.* [133], pp. 5–14.

[19] M. Kapralov and D. P. Woodruff, "Spanners and sparsifiers in dynamic streams," in *ACM Symposium on Principles of Distributed Computing, PODC '14, Paris, France, July 15-18, 2014* (M. M. Halldórsson and S. Dolev, eds.), pp. 272–281, ACM, 2014.

[20] S. Assadi, S. Khanna, Y. Li, and G. Yaroslavtsev, "Maximum matchings in dynamic graph streams and the simultaneous communication model," in *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10-12, 2016* (R. Krauthgamer, ed.), pp. 1345–1364, SIAM, 2016.

[21] S. Assadi, S. Khanna, and Y. Li, "On estimating maximum matching size in graph streams," in Klein [132], pp. 1723–1742.

[22] A. Andoni, J. Chen, R. Krauthgamer, B. Qin, D. P. Woodruff, and Q. Zhang, "On sketching quadratic forms," *Proceedings of the 2016 ACM Conference on Innovations in Theoretical Computer Science, Cambridge, MA, USA, January 14-16, 2016*, pp. 311–319, 2016.

[23] A. Jambulapati and A. Sidford, "Efficient $\tilde{O}(n/\epsilon)$ spectral sketches for the laplacian and its pseudoinverse," in *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 2487–2503, SIAM, 2018.

[24] K. J. Ahn, S. Guha, and A. McGregor, "Graph sketches: sparsification, spanners, and subgraphs," in *Proceedings of the 31st Symposium on Principles of Database Systems (PODS)*, pp. 5–14, 2012.

[25] H. Jowhari, M. Sağlam, and G. Tardos, "Tight bounds for Lp samplers, finding duplicates in streams, and related problems," in *Proceedings of the 30^{th} ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems (PODS)*, pp. 49–58, ACM, 2011.

[26] G. Cormode and D. Firmani, "A unifying framework for ℓ_0-sampling algorithms," *Distributed and Parallel Databases*, vol. 32, no. 3, pp. 315–335, 2014. Preliminary version in ALENEX 2013.

[27] M. Kapralov, J. Nelson, J. Pachocki, Z. Wang, D. P. Woodruff, and M. Yahyazadeh, "Optimal lower bounds for universal relation, and for samplers and finding duplicates in streams," in *58th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2017, Berkeley, CA, USA, October 15-17, 2017*, pp. 475–486, 2017.

[28] M. Kapralov, A. Mousavifar, C. Musco, C. Musco, and N. Nouri, "Faster spectral sparsification in dynamic streams," *CoRR*, vol. abs/1903.12165, 2019.

[29] D. A. Spielman and S.-H. Teng, "Spectral sparsification of graphs," *SIAM Journal on Computing*, vol. 40, no. 4, pp. 981–1025, 2011.

[30] D. A. Spielman and N. Srivastava, "Graph sparsification by effective resistances," in *Proceedings of the 40th Annual ACM Symposium on Theory of Computing, Victoria, British Columbia, Canada, May 17-20, 2008* (C. Dwork, ed.), pp. 563–568, ACM, 2008.

[31] I. Koutis, A. Levin, and R. Peng, "Faster spectral sparsification and numerical algorithms for SDD matrices," *ACM Trans. Algorithms*, vol. 12, no. 2, pp. 17:1–17:16, 2016.

[32] R. Kyng, Y. T. Lee, R. Peng, S. Sachdeva, and D. A. Spielman, "Sparsified cholesky and multigrid solvers for connection laplacians," in *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2016, Cambridge, MA, USA, June 18-21, 2016*, pp. 842–850, 2016.

[33] D. Cheng, Y. Cheng, Y. Liu, R. Peng, and S. Teng, "Spectral sparsification of random-walk matrix polynomials," *CoRR*, vol. abs/1502.03496, 2015.

[34] G. Jindal, P. Kolev, R. Peng, and S. Sawlani, "Density independent algorithms for sparsifying k-step random walks," in *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2017, August 16-18, 2017, Berkeley, CA, USA*, pp. 14:1–14:17, 2017.

[35] H. Li and A. Schild, "Spectral subspace sparsification," in *59th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2018, Paris, France, October 7-9, 2018*, pp. 385–396, 2018.

[36] I. Koutis, G. L. Miller, and R. Peng, "Approaching optimality for solving SDD linear systems," in *51th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2010, October 23-26, 2010, Las Vegas, Nevada, USA*, pp. 235–244, 2010.

[37] I. Koutis, G. L. Miller, and R. Peng, "A nearly-m log n time solver for SDD linear systems," in *IEEE 52nd Annual Symposium on Foundations of Computer Science, FOCS 2011, Palm Springs, CA, USA, October 22-25, 2011*, pp. 590–598, 2011.

[38] R. Peng and D. A. Spielman, "An efficient parallel solver for SDD linear systems," in *Symposium on Theory of Computing, STOC 2014, New York, NY, USA, May 31 - June 03, 2014*, pp. 333–342, 2014.

[39] M. B. Cohen, R. Kyng, G. L. Miller, J. W. Pachocki, R. Peng, A. B. Rao, and S. C. Xu, "Solving SDD linear systems in nearly $m\log^{1/2} n$ time," in *Symposium on Theory of Computing, STOC 2014, New York, NY, USA, May 31 - June 03, 2014*, pp. 343–352, 2014.

[40] R. Kyng and S. Sachdeva, "Approximate gaussian elimination for laplacians - fast, sparse, and simple," in *IEEE 57th Annual Symposium on Foundations of Computer Science, FOCS 2016, 9-11 October 2016, Hyatt Regency, New Brunswick, New Jersey, USA*, pp. 573–582, 2016.

[41] Y. T. Lee and A. Sidford, "Path finding methods for linear programming: Solving linear programs in õ(vrank) iterations and faster algorithms for maximum flow," in *55th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2014, Philadelphia, PA, USA, October 18-21, 2014*, pp. 424–433, 2014.

[42] T. Chu, Y. Gao, R. Peng, S. Sachdeva, S. Sawlani, and J. Wang, "Graph sparsification, spectral sketches, and faster resistance computation, via short cycle decompositions," in *59th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2018, Paris, France, October 7-9, 2018*, pp. 361–372, 2018.

[43] J. A. Kelner and A. Madry, "Faster generation of random spanning trees," in *50th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2009, October 25-27, 2009, Atlanta, Georgia, USA*, pp. 13–21, IEEE Computer Society, 2009.

[44] A. Madry, D. Straszak, and J. Tarnawski, "Fast generation of random spanning trees and the effective resistance metric," in *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2015, San Diego, CA, USA, January 4-6, 2015* (P. Indyk, ed.), pp. 2019–2036, SIAM, 2015.

[45] A. Schild, "An almost-linear time algorithm for uniform random spanning tree generation," in *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2018, Los Angeles, CA, USA, June 25-29, 2018* (I. Diakonikolas, D. Kempe, and M. Henzinger, eds.), pp. 214–227, ACM, 2018.

[46] V. L. Alev, N. Anari, L. C. Lau, and S. O. Gharan, "Graph clustering using effective resistance," in *9th Innovations in Theoretical Computer Science Conference, ITCS 2018, January 11-14, 2018, Cambridge, MA, USA* (A. R. Karlin, ed.), vol. 94 of *LIPIcs*, pp. 41:1–41:16, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018.

[47] A. McGregor, "Graph stream algorithms: a survey," *SIGMOD Rec.*, vol. 43, no. 1, pp. 9–20, 2014.

[48] S. Guha, A. McGregor, and D. Tench, "Vertex and hyperedge connectivity in dynamic graph streams," in *Proceedings of the 34th ACM Symposium on Principles of Database Systems, PODS 2015, Melbourne, Victoria, Australia, May 31 - June 4, 2015* (T. Milo and D. Calvanese, eds.), pp. 241–247, ACM, 2015.

[49] J. A. Kelner and A. Levin, "Spectral sparsification in the semi-streaming setting," *Theory of Computing Systems*, vol. 53, no. 2, pp. 243–262, 2013.

[50] M. B. Cohen, C. Musco, and J. Pachocki, "Online row sampling," in *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2016)*, 2016.

[51] R. Kyng, J. Pachocki, R. Peng, and S. Sachdeva, "A framework for analyzing resparsification algorithms," in *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 2032–2043, 2017.

[52] I. Abraham, D. Durfee, I. Koutis, S. Krinninger, and R. Peng, "On fully dynamic graph sparsifiers," in *57th Annual Symposium on Foundations of Computer Science*, pp. 335–344, 2016.

[53] D. A. Spielman and S. Teng, "Spectral sparsification of graphs," *SIAM J. Comput.*, vol. 40, no. 4, pp. 981–1025, 2011.

[54] J. D. Batson, D. A. Spielman, and N. Srivastava, "Twice-ramanujan sparsifiers," in *Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC 2009, Bethesda, MD, USA, May 31 - June 2, 2009* (M. Mitzenmacher, ed.), pp. 255–262, ACM, 2009.

[55] Z. Allen Zhu, Z. Liao, and L. Orecchia, "Spectral sparsification and regret minimization beyond matrix multiplicative updates," in Servedio and Rubinfeld [134], pp. 237–245.

[56] Y. T. Lee and H. Sun, "Constructing linear-sized spectral sparsification in almost-linear time," in *IEEE 56th Annual Symposium on Foundations of Computer Science, FOCS 2015, Berkeley, CA, USA, 17-20 October, 2015* (V. Guruswami, ed.), pp. 250–269, IEEE Computer Society, 2015.

[57] Y. T. Lee and H. Sun, "An sdp-based algorithm for linear-sized spectral sparsification," in *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2017, Montreal, QC, Canada, June 19-23, 2017* (H. Hatami, P. McKenzie, and V. King, eds.), pp. 678–687, ACM, 2017.

[58] P. Indyk and R. 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* (J. S. Vitter, ed.), pp. 604–613, ACM, 1998.

[59] M. Datar, N. Immorlica, P. Indyk, and V. S. Mirrokni, "Locality-sensitive hashing scheme based on p-stable distributions," in *Proceedings of the 20th ACM Symposium on Computational Geometry, Brooklyn, New York, USA, June 8-11, 2004* (J. Snoeyink and J. Boissonnat, eds.), pp. 253–262, ACM, 2004.

[60] A. Andoni and P. Indyk, "Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions," in *47th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2006), 21-24 October 2006, Berkeley, California, USA, Proceedings* [131], pp. 459–468.

[61] A. Andoni, P. Indyk, H. L. Nguyen, and I. P. Razenshteyn, "Beyond locality-sensitive hashing," in *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2014, Portland, Oregon, USA, January 5-7, 2014* (C. Chekuri, ed.), pp. 1018–1028, SIAM, 2014.

[62] A. Andoni and I. P. Razenshteyn, "Optimal data-dependent hashing for approximate near neighbors," in Servedio and Rubinfeld [134], pp. 793–801.

[63] W. Hoeffding, "Probability inequalities for sums of bounded random variables," *Journal of the American Statistical Association*, vol. 58, no. 301, pp. 13–30, 1963.

[64] J. A. Kelner, L. Orecchia, A. Sidford, and Z. Allen Zhu, "A simple, combinatorial algorithm for solving SDD systems in nearly-linear time," in *Symposium on Theory of Computing Conference, STOC'13, Palo Alto, CA, USA, June 1-4, 2013*, pp. 911–920, 2013.

[65] Y. T. Lee and A. Sidford, "Efficient accelerated coordinate descent methods and faster algorithms for solving linear systems," in *54th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2013, 26-29 October, 2013, Berkeley, CA, USA*, pp. 147–156, 2013.

[66] A. Filtser, M. Kapralov, and N. Nouri, "Graph spanners by sketching in dynamic streams and the simultaneous communication model," in *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms, SODA 2021, Virtual Conference, January 10 - 13, 2021* (D. Marx, ed.), pp. 1894–1913, SIAM, 2021.

[67] K. J. Ahn, S. Guha, and A. McGregor, "Analyzing graph structure via linear measurements," in Rabani [130], pp. 459–467.

[68] K. J. Ahn, S. Guha, and A. McGregor, "Graph sketches: sparsification, spanners, and subgraphs," in Benedikt *et al.* [133], pp. 5–14.

[69] K. J. Ahn, S. Guha, and A. McGregor, "Spectral sparsification in dynamic graph streams," in *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques - 16th International Workshop, APPROX 2013, and 17th International Workshop, RANDOM 2013, Berkeley, CA, USA, August 21-23, 2013. Proceedings* (P. Raghavendra, S. Raskhodnikova, K. Jansen, and J. D. P. Rolim, eds.), vol. 8096 of *Lecture Notes in Computer Science*, pp. 1–10, Springer, 2013.

[70] A. Andoni, J. Chen, R. Krauthgamer, B. Qin, D. P. Woodruff, and Q. Zhang, "On sketching quadratic forms," in *Proceedings of the 2016 ACM Conference on Innovations in Theoretical Computer Science*, ITCS '16, (New York, NY, USA), p. 311–319, Association for Computing Machinery, 2016.

[71] Y. Li, H. L. Nguyen, and D. P. Woodruff, "Turnstile streaming algorithms might as well be linear sketches," in *Symposium on Theory of Computing, STOC 2014, New York, NY, USA, May 31 - June 03, 2014* (D. B. Shmoys, ed.), pp. 174–183, ACM, 2014.

[72] K. Hosseini, S. Lovett, and G. Yaroslavtsev, "Optimality of linear sketching under modular updates," in *34th Computational Complexity Conference, CCC 2019, July 18-20, 2019, New Brunswick, NJ, USA* (A. Shpilka, ed.), vol. 137 of *LIPIcs*, pp. 13:1–13:17, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019.

[73] S. Kannan, E. Mossel, S. Sanyal, and G. Yaroslavtsev, "Linear sketching over f_2," in *33rd Computational Complexity Conference, CCC 2018, June 22-24, 2018, San Diego, CA, USA* (R. A. Servedio, ed.), vol. 102 of *LIPIcs*, pp. 8:1–8:37, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018.

[74] J. Kallaugher and E. Price, "Separations and equivalences between turnstile streaming and linear sketching," in *Proccedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing, STOC 2020, Chicago, IL, USA, June 22-26, 2020* (K. Makarychev, Y. Makarychev, M. Tulsiani, G. Kamath, and J. Chuzhoy, eds.), pp. 1223–1236, ACM, 2020.

[75] M. Elkin and S. Solomon, "Fast constructions of lightweight spanners for general graphs," *ACM Trans. Algorithms*, vol. 12, no. 3, pp. 29:1–29:21, 2016. See also SODA'13.

[76] S. Alstrup, S. Dahlgaard, A. Filtser, M. Stöckel, and C. Wulff-Nilsen, "Constructing light spanners deterministically in near-linear time," in *27th Annual European Symposium on Algorithms, ESA 2019, September 9-11, 2019, Munich/Garching, Germany.* (M. A. Bender, O. Svensson, and G. Herman, eds.), vol. 144 of *LIPIcs*, pp. 4:1–4:15, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019. Full version: https://arxiv.org/abs/1709.01960.

[77] I. Althöfer, G. Das, D. P. Dobkin, D. Joseph, and J. Soares, "On sparse spanners of weighted graphs," *Discret. Comput. Geom.*, vol. 9, pp. 81–100, 1993.

[78] A. Filtser and S. Solomon, "The greedy spanner is existentially optimal," *SIAM J. Comput.*, vol. 49, no. 2, pp. 429–447, 2020.

[79] S. Baswana and S. Sen, "A simple and linear time randomized algorithm for computing sparse spanners in weighted graphs," *Random Struct. Algorithms*, vol. 30, no. 4, pp. 532–563, 2007.

[80] J. Kallaugher, A. McGregor, E. Price, and S. Vorotnikova, "The complexity of counting cycles in the adjacency list streaming model," in *Proceedings of the 38th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, PODS 2019, Amsterdam, The Netherlands, June 30 - July 5, 2019* (D. Suciu, S. Skritek, and C. Koch, eds.), pp. 119–133, ACM, 2019.

[81] J. Kallaugher, M. Kapralov, and E. Price, "The sketching complexity of graph and hypergraph counting," in *59th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2018, Paris, France, October 7-9, 2018* (M. Thorup, ed.), pp. 556–567, IEEE Computer Society, 2018.

[82] R. Ahmed, G. Bodwin, F. D. Sahneh, K. Hamm, M. J. Latifi Jebelli, S. Kobourov, and R. Spence, "Graph spanners: A tutorial review," *Computer Science Review*, vol. 37, p. 100253, 2020.

[83] M. Elkin, "Streaming and fully dynamic centralized algorithms for constructing and maintaining sparse spanners," *ACM Trans. Algorithms*, vol. 7, no. 2, pp. 20:1–20:17, 2011.

[84] S. Baswana, "Streaming algorithm for graph spanners - single pass and constant processing time per edge," *Inf. Process. Lett.*, vol. 106, no. 3, pp. 110–114, 2008.

[85] S. Baswana, S. Khurana, and S. Sarkar, "Fully dynamic randomized algorithms for graph spanners," *ACM Trans. Algorithms*, vol. 8, no. 4, pp. 35:1–35:51, 2012.

[86] A. Bernstein, S. Forster, and M. Henzinger, "A deamortization approach for dynamic spanner and dynamic maximal matching," in *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019, San Diego, California, USA, January 6-9, 2019*, pp. 1899–1918, 2019.

[87] M. Kapralov and R. Panigrahy, "Spectral sparsification via random spanners," in *Innovations in Theoretical Computer Science 2012, Cambridge, MA, USA, January 8-10, 2012* (S. Goldwasser, ed.), pp. 393–398, ACM, 2012.

[88] I. Koutis and S. C. Xu, "Simple parallel and distributed algorithms for spectral graph sparsification," *ACM Trans. Parallel Comput.*, vol. 3, no. 2, pp. 14:1–14:14, 2016.

[89] M. Kapralov, N. Nouri, A. Sidford, and J. Tardos, "Dynamic streaming spectral sparsification in nearly linear time and space," *CoRR*, vol. abs/1903.12150, 2019.

[90] M. Kapralov, A. Mousavifar, C. Musco, C. Musco, and N. Nouri, "Faster spectral sparsification in dynamic streams," *CoRR*, vol. abs/1903.12165, 2019.

[91] M. Charikar, M. Kapralov, N. Nouri, and P. Siminelakis, "Kernel density estimation through density constrained near neighbor search," in *61st IEEE Annual Symposium on Foundations of Computer Science, FOCS 2020, Durham, NC, USA, November 16-19, 2020* (S. Irani, ed.), pp. 172–183, IEEE, 2020.

[92] J. Fan and I. Gijbels, *Local polynomial modelling and its applications: monographs on statistics and applied probability 66*, vol. 66. CRC Press, 1996.

[93] B. Scholkopf and A. J. Smola, *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT press, 2001.

[94] S. Joshi, R. V. Kommaraji, J. M. Phillips, and S. Venkatasubramanian, "Comparing distributions and shapes using the kernel distance," in *Proceedings of the twenty-seventh annual symposium on Computational geometry*, pp. 47–56, ACM, 2011.

[95] E. Schubert, A. Zimek, and H.-P. Kriegel, "Generalized outlier detection with flexible kernel density estimates," in *Proceedings of the 2014 SIAM International Conference on Data Mining*, pp. 542–550, SIAM, 2014.

[96] C. R. Genovese, M. Perone-Pacifico, I. Verdinelli, L. Wasserman, *et al.*, "Nonparametric ridge estimation," *The Annals of Statistics*, vol. 42, no. 4, pp. 1511–1545, 2014.

[97] E. Arias-Castro, D. Mason, and B. Pelletier, "On the estimation of the gradient lines of a density and the consistency of the mean-shift algorithm," *Journal of Machine Learning Research*, 2015.

[98] E. Gan and P. Bailis, "Scalable kernel density classification via threshold-based pruning," in *Proceedings of the 2017 ACM International Conference on Management of Data*, pp. 945–959, ACM, 2017.

[99] J. Shawe-Taylor, N. Cristianini, *et al.*, *Kernel methods for pattern analysis*. Cambridge university press, 2004.

[100] C. E. Rasmussen and C. K. I. Williams, *Gaussian processes for machine learning*. Adaptive computation and machine learning, MIT Press, 2006.

[101] R. Beatson and L. Greengard, *A short course on fast multipole methods*, pp. 1–37. Numerical Mathematics and Scientific Computation, Oxford University Press, 1997.

[102] A. G. Gray and A. W. Moore, "N-body'problems in statistical learning," in *Advances in neural information processing systems*, pp. 521–527, 2001.

[103] A. G. Gray and A. W. Moore, "Nonparametric density estimation: Toward computational tractability," in *Proceedings of the 2003 SIAM International Conference on Data Mining*, pp. 203–211, SIAM, 2003.

[104] C. Yang, R. Duraiswami, N. A. Gumerov, and L. Davis, "Improved fast gauss transform and efficient kernel density estimation," in *Proceedings of the Ninth IEEE International Conference on Computer Vision-Volume 2*, p. 464, IEEE Computer Society, 2003.

[105] D. Lee, A. W. Moore, and A. G. Gray, "Dual-tree fast gauss transforms," in *Advances in Neural Information Processing Systems*, pp. 747–754, 2006.

[106] P. Ram, D. Lee, W. March, and A. G. Gray, "Linear-time algorithms for pairwise statistical problems," in *Advances in Neural Information Processing Systems*, pp. 1527–1535, 2009.

[107] A. Backurs, M. Charikar, P. Indyk, and P. Siminelakis, "Efficient density evaluation for smooth kernels," in *2018 IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS)*, pp. 615–626, IEEE, 2018.

[108] A. Backurs, P. Indyk, and T. Wagner, "Space and time efficient kernel density estimation in high dimensions," in *Advances in Neural Information Processing Systems*, 2019.

[109] A. Rubinstein, "Hardness of approximate nearest neighbor search," in *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing*, pp. 1260–1268, 2018.

[110] D. R. Karger and M. Ruhl, "Finding nearest neighbors in growth-restricted metrics," in *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*, pp. 741–750, ACM, 2002.

[111] M. Datar, N. Immorlica, P. Indyk, and V. S. Mirrokni, "Locality-sensitive hashing scheme based on p-stable distributions," in *Proceedings of the twentieth annual symposium on Computational geometry*, pp. 253–262, ACM, 2004.

[112] P. Syminelakis, *Kernel Evaluation in High Dimensions: Importance Sampling and Nearest-Neighbor Search*. PhD thesis, Stanford University, 2019.

[113] P. Siminelakis, K. Rong, P. Bailis, M. Charikar, and P. Levis, "Rehashing kernel evaluation in high dimensions," in *International Conference on Machine Learning*, pp. 5789–5798, 2019.

[114] R. Spring and A. Shrivastava, "A new unbiased and efficient class of lsh-based samplers and estimators for partition function computation in log-linear models," *arXiv preprint arXiv:1703.05160*, 2017.

[115] C. Luo and A. Shrivastava, "Arrays of (locality-sensitive) count estimators (ace): Anomaly detection on the edge," in *Proceedings of the 2018 World Wide Web Conference*, pp. 1439–1448, International World Wide Web Conferences Steering Committee, 2018.

[116] B. Chen, Y. Xu, and A. Shrivastava, "Lsh-sampling breaks the computation chicken-and-egg loop in adaptive stochastic gradient estimation," *arXiv preprint arXiv:1910.14162*, 2019.

[117] C. Luo and A. Shrivastava, "Scaling-up split-merge mcmc with locality sensitive sampling (lss)," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, pp. 4464–4471, 2019.

[118] X. Wu, M. Charikar, and V. Natchu, "Local density estimation in high dimensions," in *International Conference on Machine Learning*, pp. 5293–5301, 2018.

[119] J. M. Phillips, "ε-samples for kernels," in *Proceedings of the twenty-fourth annual ACM-SIAM symposium on Discrete algorithms*, pp. 1622–1632, SIAM, 2013.

[120] J. M. Phillips and W. M. Tai, "Improved coresets for kernel density estimates," in *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 2718–2727, SIAM, 2018.

[121] J. M. Phillips and W. M. Tai, "Near-optimal coresets of kernel density estimates," in *34th International Symposium on Computational Geometry (SoCG 2018)*, Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2018.

[122] Z. S. Karnin and E. Liberty, "Discrepancy, coresets, and sketches in machine learning," in *Conference on Learning Theory, COLT 2019, 25-28 June 2019, Phoenix, AZ, USA* (A. Beygelzimer and D. Hsu, eds.), vol. 99 of *Proceedings of Machine Learning Research*, pp. 1975–1993, PMLR, 2019.

[123] H. Avron, H. Nguyen, and D. Woodruff, "Subspace embeddings for the polynomial kernel," in *Advances in Neural Information Processing Systems 27* (Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, eds.), pp. 2258–2266, Curran Associates, Inc., 2014.

[124] N. Pham and R. Pagh, "Fast and scalable polynomial kernels via explicit feature maps," in *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 239–247, ACM, 2013.

[125] H. Avron, M. Kapralov, C. Musco, C. Musco, A. Velingker, and A. Zandieh, "Random fourier features for kernel ridge regression: Approximation bounds and statistical guarantees," in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 253–262, JMLR. org, 2017.

[126] T. D. Ahle, M. Kapralov, J. B. Knudsen, R. Pagh, A. Velingker, D. P. Woodruff, and A. Zandieh, "Oblivious sketching of high-degree polynomial kernels," in *SODA (to appear)*, 2020.

[127] M. Li, G. L. Miller, and R. Peng, "Iterative row sampling," in *54th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2013, 26-29 October, 2013, Berkeley, CA, USA*, pp. 127–136, IEEE Computer Society, 2013.

[128] D. Achlioptas, "Database-friendly random projections: Johnson-lindenstrauss with binary coins," *J. Comput. Syst. Sci.*, vol. 66, no. 4, pp. 671–687, 2003.

[129] M. Kapralov, J. Nelson, J. Pachocki, Z. Wang, D. P. Woodruff, and M. Yahyazadeh, "Optimal lower bounds for universal relation, and for samplers and finding duplicates in streams," in *58th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2017, Berkeley, CA, USA, October 15-17, 2017* (C. Umans, ed.), pp. 475–486, IEEE Computer Society, 2017.

[130] Y. Rabani, ed., *Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2012, Kyoto, Japan, January 17-19, 2012*, SIAM, 2012.

[131] *47th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2006), 21-24 October 2006, Berkeley, California, USA, Proceedings*, IEEE Computer Society, 2006.

[132] P. N. Klein, ed., *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2017, Barcelona, Spain, Hotel Porta Fira, January 16-19*, SIAM, 2017.

[133] M. Benedikt, M. Krötzsch, and M. Lenzerini, eds., *Proceedings of the 31st ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS 2012, Scottsdale, AZ, USA, May 20-24, 2012*, ACM, 2012.

[134] R. A. Servedio and R. Rubinfeld, eds., *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing, STOC 2015, Portland, OR, USA, June 14-17, 2015*, ACM, 2015.

Navid Nouri

Curriculum Vitae

School of Computer and Communication Sciences
EPFL
(+41) 78 730 75 18
navidnouri1992@gmail.com
www.linkedin.com/in/navid-nouri-b6587a1b8/

Education

2017–2022 **PhD in Computer Science, (Advisor: Prof. Michael Kapralov)**, *EPFL*, Lausanne, Switzerland
My research was focused on designing and implementing efficient algorithms and data structures for problems with applications in machine learning such as Kernel Density Estimation, Near Neighbor Search, Graph Sparsification and Graph Spanners.

2011–2016 **B.Sc. in Electrical Engineering (Communications), Minoring in Mathematical Sciences**, *Sharif University of Technology*, Tehran, Iran

Work Experience

Summer 2022 (3 months) **Internship**, **JPMorgan Chase & Co. (AI research team)**, London, the United Kingdom
I was honored to be a part of the AI Research and synthetic data team. As an AI research intern, I had the opportunity to conduct research under the supervision of Dr. Eleonora Kreacic and Dr. Vamsi Potluru. In this period, we were able to:
- design explainable differentially private synthetic dataset generation algorithms with theoretical guarantees
- optimize a practical version of the algorithm to perform faster on real-world datasets with a large number of rows and features
- test the quality of output synthetic datasets with various metrics such as KS test via SDV library and compare it to existing synthesizers
- prepare the manuscript of this result to be submitted for publication

Summer 2021 (6 months, from mid-June to mid-Dec) **Internship**, **Credit Suisse (International Wealth Management)**, Zurich, Switzerland
I was fortunate to be a part of Wealth Engine team and collaborate with highly skilled and professional individuals in the team. During this internship, I was able to:
- work end-to-end on a natural language processing project
- work as a machine learning engineer on the ML aspect of the project
- integrate the project in a micro-service architecture
- get involved in discussions with business stakeholders and implement their needs

Summer 2014 and 2015 **Internship (Junior Research Assistant)**, *Institute of Network Coding*, Chinese University of Hong Kong, Hong Kong
In these two internships, I was fortunate to work under the supervision of Prof. Chung Chan.
- In the first internship, I was working as a junior member of a group of highly skilled and knowledgeable people. Mainly, my task was to bridge the gap between two related research areas for this group. As my first international experience, I gained wonderful experiences in a professional environment.
- In the second internship, I was also working on a problem of designing low communication schemes in graph networks. I had the privilege of working with a group of highly motivated people from various cultures, which enriched my communication skills.

Skills

Programming Languages **Python (Libraries: Numpy, Pandas, Matplotlib, Scikit-learn, FastAPI, JSON, PyTest,...)**
- **Hands-on experience using Python and its libraries during my internship at Credit Suisse.**

C++, Java, TypeScript, MATLAB, R

Other **Linux (skilled in using terminal and bash scripting)**
Git, Angular (TypeScript, CSS, HTML), Docker, OpenShift, REST-API,
- **Hands-on experience using these tools during my internship at Credit Suisse.**

mySQL, LaTeX, Microsoft Word, Excel, PowerPoint

Honors and Awards

2011 **Ranked 19^{th} in Iran's University Entrance Exam, among 250,000 test-takers**

2011-2016 **Fellowship of Iran's National Elite Foundation**

2009 **Honorary Diploma in Tournament of Towns international Mathematical Contest**

2008 **Ranked 1^{st} in Paya National Mathematical Contest among more than 3000 groups**

Publications

NeurIPS 2021 **Efficient and Local Parallel Random Walks**,
With Michael Kapralov, Silvio Lattanzi and Jakab Tardos,
(ArXiv link: https://arxiv.org/pdf/2112.00655.pdf)

SODA 2021 **Graph Spanners by Sketching in Dynamic Streams and the Simultaneous Communication Model**,
With Arnold Filtser and Michael Kapralov,
(ArXiv link: https://arxiv.org/abs/2007.14204)

FOCS 2020 **Kernel Density Estimation through Density Constrained Near Neighbor Search**,
With Moses Charikar, Michael Kapralov and Paris Syminelakis,
(ArXiv link: https://arxiv.org/abs/2011.06997)

AISTATS 2020 **Scaling up Kernel Ridge Regression via Locality Sensitive Hashing**,
With Michael Kapralov, Ilya Razenshteyn, Ameya Velingker and Amir Zandieh,
(ArXiv link: https://arxiv.org/abs/2003.09756)

SODA 2020 **Fast and Space Efficient Spectral Sparsification in Dynamic Streams**,
With Michael Kapralov, Aida Mousavifar, Cameron Musco, Christopher Musco, Aaron Sidford and Jakab Tardos, This paper is the merger of the following papers:

- Faster Spectral Sparsification in Dynamic Streams
  With Michael Kapralov, Aida Mousavifar, Cameron Musco and Christopher Musco,
  (ArXiv link: https://arxiv.org/abs/1903.12165)
- Dynamic Streaming Spectral Sparsification in Nearly Linear Time and Space
  With Michael Kapralov, Aaron Sidford and Jakab Tardos,
  (ArXiv link: https://arxiv.org/abs/1903.12150)

Manuscripts

Submitted for review and available on Arxiv **Age of Information-Reliability Trade-offs in Energy Harvesting Sensor Networks**,
With Darya Ardan and Mahmood Mohassel Feghhi,
(ArXiv link: https://arxiv.org/abs/2008.00987)

Teaching Experiences

2020 **Teaching Assistant of Prof. Kapralov**, *Advanced Algorithms*

2019 **Teaching Assistant of Prof. Svensson**, *Advanced Algorithms*

2018, 2019 **Teaching Assistant of Prof. Kapralov**, *Algorithms*

2018 **Teaching Assistant of Prof. Anthony C. Davison**, *Probability and Statistics*

Graduate Courses' Grades

Sublinear Algorithms for Big Data Analysis, *Prof. Kapralov*, 6/6

Advanced Algorithms, *Prof. Svensson*, 6/6

Information Theory and Coding , *Prof. Telatar*, 6/6

Language Skills

English Professional Working Proficiency

Turkish Fluent

Azeri Native (Mother Tongue)

Persian Native