Nearest Neighbor Search: the Old, the New, and the Impossible

by

Alexandr Andoni

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 2009

Author . . . .
Department of Electrical Engineering and Computer Science
September 4, 2009

Certified by . . . .
Piotr Indyk
Associate Professor
Thesis Supervisor

Accepted by . . . .
Terry P. Orlando
Chairman, Department Committee on Graduate Students

Nearest Neighbor Search:
the Old, the New, and the Impossible
by
Alexandr Andoni

Submitted to the Department of Electrical Engineering and Computer Science
on September 4, 2009, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

Abstract

Over the last decade, an immense amount of data has become available. From collections of photos, to genetic data, and to network traffic statistics, modern technologies and cheap storage have made it possible to accumulate huge datasets. But how can we effectively use all this data? The ever growing sizes of the datasets make it imperative to design new algorithms capable of sifting through this data with extreme efficiency.

A fundamental computational primitive for dealing with massive dataset is the Nearest Neighbor (NN) problem. In the NN problem, the goal is to preprocess a set of objects, so that later, given a query object, one can find efficiently the data object most similar to the query. This problem has a broad set of applications in data processing and analysis. For instance, it forms the basis of a widely used classification method in machine learning: to give a label for a new object, find the most similar labeled object and copy its label. Other applications include information retrieval, searching image databases, finding duplicate files and web pages, vector quantization, and many others.

To represent the objects and the similarity measures, one often uses geometric notions. For example, a black-and-white image may be modeled by a high-dimensional vector, with one coordinate per pixel, whereas the similarity measure may be the standard Euclidean distance between the resulting vectors. Many other, more elaborate ways of representing objects by high-dimensional feature vectors have been studied.

In this thesis, we study the NN problem, as well as other related problems that occur frequently when dealing with the massive datasets. Our contribution is two-fold: we significantly improve the algorithms within the classical approaches to NN, as well as propose new approaches where the classical ones fail. We focus on several key distances and similarity measures, including the Euclidean distance, string edit distance and the Earth-Mover Distance (a popular method for comparing images). We also give a number of impossibility results, pointing out the limits of the NN algorithms.

The high-level structure of our thesis is summarized as follows.

New algorithms via the classical approaches. We give a new algorithm for the approximate NN problem in the d-dimensional Euclidean space. For an approximation factor $c > 1$, our algorithm achieves dn^{ρ} query time and $dn^{1+\rho}$ space for $\rho = 1/c^2 + o(1)$. This greatly improves on the previous algorithms that achieved ρ that was only slightly smaller than $1/c$. The same technique also yields an algorithm with $dn^{O(\rho)}$ query time and space near-linear in n. Furthermore, our algorithm is *near-optimal* in the class of "hashing" algorithms.

Failure of the classical approaches for some hard distances. We give an evidence that the classical approaches to NN under certain hard distances, such as the string edit

distance, meet a concrete barrier at a nearly logarithmic approximation. Specifically, we show that for all classical approaches to NN under the edit distance, involving embeddings into a general class of spaces (such as ℓ_1, powers of ℓ_2, etc), the resulting approximation has to be at least near-logarithmic in the strings' length.

A new approach to NN under hard distances. Motivated by the above impossibility results, we develop a new approach to the NN problem, where the classical approaches fail. Using this approach, we give a new efficient NN algorithm for a variant of the edit distance, the Ulam distance, which achieves a double-logarithmic approximation. This is an exponential improvement over the *lower bound* on the approximation achievable via the previous classical approaches to this problem.

Data structure lower bounds. To complement our algorithms, we prove lower bounds on NN data structures for the Euclidean distance and for the mysterious but important case of the ℓ_∞ distance. In both cases, our lower bounds are the *first ones* to hold in the same computational model as the respective upper bounds. Furthermore, for both problems, our lower bounds are *optimal* in the considered models.

External applications. Although our main focus is on the NN problem, our techniques naturally extend to related problems. We give such applications for each of our algorithmic tools. For example, we give an algorithm for computing the edit distance between two strings of length d in near-linear time. Our algorithm achieves approximation $2^{\tilde{O}(\sqrt{\log d})}$, improving over the previous bound of $d^{1/3+o(1)}$. We note that this problem has a classical exact algorithm based on dynamic programming, running in quadratic time.

Thesis Supervisor: Piotr Indyk
Title: Associate Professor

Acknowledgments

For the existence of this thesis, I owe a huge "Thank you" to many people. The hardest of all it is to express the entire extent of my gratitude to Piotr Indyk, my adviser. He combined all the right ingredients — including incessant support, genuine interest, and humor — to make my experience most enjoyable and complete. Reflecting on the past years, I am only impressed how he has always seemed to know precisely what to do in order to keep me captivated and enthusiastic about research. Thanks to his effort, not only have I reached my personal aspirations, but have seen what lies beyond them.

I met Piotr in 2002, when I was an undergraduate student at MIT. In the summer of that year, he gave me a cute problem: prove a lower bound for embedding edit distance into normed spaces. At that time, I would only understand the "edit distance" part of the problem formulation, but that was enough for me. It was easy to like the problem: it seemed important with tangible motivation, and admitted some partial-progress solution that was non-trivial but still within reach. The project worked out well: in several weeks we had a result, later accepted to SODA. Moreover, 4.5 years later, I returned to the problem, and an improved bound is now part of the present thesis. This story is a perfect illustration of Piotr's exemplary advising style: get me interested with just the right combination of motivation, projection of personal enthusiasm, challenge, collaboration, and then let me explore. For putting all this tremendous effort and more, I will always remain in debt to him.

I would also like to thank Robi Krauthgamer and Ronitt Rubinfeld, who, in many ways, have been my informal advisers. I spoke to Robi first time at SODA'06, when I asked him about an internship opportunity at IBM Almaden. While he encouraged me to apply, he also mentioned that it is very competitive to obtain the internship. I was extremely lucky that it wasn't *too* competitive, and that we started our years-long collaboration that summer. Many of the results from this thesis have been developed in collaboration with him. I thank Robi for hosting me at IBM Almaden and Weizmann Institute, for being a brilliant and responsible collaborator, and, the last but not the least, for being a great friend.

Ronitt has influenced me in many ways through advising on matters related to research, and academic life in general. I have learned from and enjoyed all my interactions with her, including collaborating on the two joint papers, co-organizing the theory colloquium, and teaching a class. I very much appreciated her personal, warm style which has always been refreshing, and brought me back from the steel world of razor-thin process of matching the upper and lower bounds to the world of human researchers doing what they like most, namely discovering exciting new facts (which could very well be the *delightfully* matching upper and lower bounds!).

My summer internships at IBM Almaden and MSR SVC were of immense importance in my formation as a researcher. They provided the much desired "non-MIT" research exposure, and, above all, an opportunity to meet and collaborate with some outstanding people in a stellar environment. For this, I thank the IBM Almaden group, with special thanks to Ken Clarkson, Jayram, and Ron Fagin, as well as the MSR SVC group, with special thanks to Rina Panigrahy, Udi Wieder, Kunal Talwar, and Cynthia Dwork.

My research has been greatly influenced by my collaborators: Anupam, Avinatan, Costis, David, Dorian, Huy, Jayram, Jessica, Kevin, Khanh, Krzysztof, Mayur, Michel, Mihai, Nicole, Ning, Noga, Piotr, Robi, Ronitt, Sebastien, Sofya, Tali, and Vahab. Thank you all, I have learned a lot from you. To many I am thankful for also being much more than "just" collaborators.

A number of my friends have been the spine of my entire graduate student life (of both the "graduate student" and the "life" components). Without them I do not know how I could have fared this far: their optimism, enthusiasm, and support have been an inspiration. My old friend, Mihai, has always been great to talk to, on academic and non-academic topics, and party with (mostly on non-academic topics), which made him a superb office-mate and apartment-mate. I benefited a lot from his questions that would immediately pierce to the heart of the matter. My friends Krzysztof and David were of invaluable help at so many stages and moments of my PhD years, that I am lost at finding a place to start. I hope it suffices to mention that they were friends I could rely on for 100% of the time: in good and bad times, as well as in my productive and my unproductive periods. To Krzysztof, I am also much indebted for reading this thesis. For a host of amicable support I also owe a lot to Jelani, Bernhard, Olek and Kamila.

My non-academic friends were responsible for keeping me in contact with the real life and relative sanity. I am very happy to have Dumi, Cristi, Daniela, Bogdan, Tudor, Dorian, and Corina as my very close friends.

I would also like to thank the students from the MIT theory group for creating an inspiring, dynamic, and vibrant atmosphere in the lab. These include (in no particular order) Tasos, Nick, Sergey, Edi, Arnab, Ning, Kevin, Huy, Ben, Guy, MinJi, Rotem, Alex, Khanh, Avinatan, Swastik, Shubhangi, Ankur, Petar, Elena, Nadia, Oren, Szymon, and Jacob. I also thank Madhu Sudan and Silvio Micali for the rare but very insightful discussions about many academic matters. I wish I knew to tap into these amazingly immense resources earlier and more often.

I extend my thanks to my teachers and mentors from the high school, for all the admirable work they have put into teaching me the love of knowledge, and for their continuous believing into me. They have gone out of their way to give me the right qualities, which later brought me to MIT.

There have been four years for me to think what to write in this acknowledgments. Yet, I still wrote it on the night before the deadline. I apologize to all whom I forgot to properly thank here. Thank you all!

Above all, my greatest gratitude I devote to my family, who laid the foundations of all of this for me. The seeds of their contribution have always permeated my entire life and will continue to do so. No words will be enough to express all my appreciation for their support and everything they have done for me.

To my brother Alexei for inspiring my curiosity.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Modern technologies have made it possible to collect vast amounts of data, such as media data (images, music, videos), network information data (Internet, network monitoring), genomic data, medical data (personal biometrics), astronomical data (Sloan digital sky map, SETI data), etc. Now that we have this data, terabytes and petabytes of it, what do we do with it?

Broadly speaking, we would like to extract “useful information" from the data. For example, consider a collection of images (say, from `Flickr.com`). A basic and useful goal would be to label the images by what they contain (e.g., ‘a tiger’, ‘an elephant’, ‘a human’, etc). Without any prior information, this would have been quite a difficult task. However, if we had a subset of the database that was labeled already, e.g., by hand, then we could deduce the label of each new image simply by finding the most “similar” labeled image and copying its label.

In order to accomplish this task, we need to solve the following computational problem: given a collection of objects (such as images), for each unlabeled object q, find the labeled object p that is most similar to q, under some notion of similarity. This problem, often called the *all-pairs-nearest-neighbor problem*, admits a straightforward solution: just consider all possible pairs of labeled and unlabeled objects and check how similar they are. However, this approach is not feasible for modern datasets: even merely enumerating all pairs from a collection of, say, *billions* of images would take too much time!

With the datasets only getting larger and more ubiquitous (e.g., see Table 1.1), it becomes imperative to design algorithms capable of processing the data extremely efficiently. Ideally, we would like algorithms with runtimes that are at most linear in the size of the dataset, as opposed to, say, quadratic as above. This goal often requires us to reformulate the algorithm designer’s question from “can we solve a problem in polynomial time?” to “what can we do in near-linear time?”

Dataset	**Size**	**Reference**
NCBI dataset, genomes	over 193 billion bases	[NCB]
`Flickr`, photos	over 3 billion photos	[Fli]
`Facebook`, photos	over 1 billion photos uploaded per month	[Fac]
`Akamai`, HTTP hits	over 4 million per second	[Aka]

Table 1.1: Sizes of example datasets.

What are the basic computational questions that we want to solve on these datasets? A fundamental primitive that regularly emerges is the *nearest neighbor (NN) problem*[1]. The problem is defined as follows: given a collection of n objects, build a data structure which, given arbitrary query object, reports the dataset object that is most similar to the query.

The motivation for this problem is multifold. First, it represents an "online version" of the aforementioned all-pairs-nearest-neighbor problem. Specifically, one can solve the latter problem by constructing a NN data structure on the labeled objects, and then running a NN query for each of the unlabeled objects. In fact, in *all* scenarios considered in this thesis, this approach provides the *best known* algorithm for the all-pairs-nearest-neighbor problem.

More broadly, the NN problem is of major importance in a number application areas, including data compression, databases and data mining, information retrieval, searching image datasets, machine learning, pattern recognition, statistics, and data analysis. For example, the "nearest neighbor rule" is perhaps one of the most basic machine learning classification rules. In fact, our example from above is precisely an implementation of this rule. To give another example, suppose that a genome of a new organism is sequenced, and one wants to find genes that are similar to those in the existing databases. This again corresponds to the NN problem, with the *edit distance*[2] as the measure of (dis)similarity. In fact, such a service is already provided to the public by the National Center for Biotechnology Information (NCBI), under the auspices of NIH [NCB].

In order to define the NN problem, we need to specify the representation of the objects as well as the similarity measures. Typically, the features of each object of interest (such as an image) are represented as a point in a *high-dimensional* space $\mathbb{R}^d$ and the similarity is measured using a distance metric. The number of features (i.e., the dimensionality) ranges anywhere from tens to millions. For example, one can represent a 1000×1000 image as a vector in a $1,000,000$-dimensional space, one dimension per pixel. Then, we can approximate the dissimilarity between two such objects by the Euclidean distance between the corresponding points. Many other, more advanced, high-dimensional representations of images and dissimilarity measures are known; we will cover them in this thesis as well.

1.1 Curse of Dimensionality: Diagnosis and Cure

A particularly natural and well-studied instance of the NN problem is in the Euclidean space, where the data points live in a d-dimensional space $\mathbb{R}^d$ under the Euclidean distance function, where dimension d may range up to hundreds or even thousands.

A number of classical algorithms are known for the case when d is very low. When $d = 1$, i.e., when the points are on a real line, the NN problem becomes the classical *predecessor* problem. A basic solution is to sort the points, and then, during a query time, perform a binary search. This already achieves $O(n)$ space and $O(\log n)$ query time.

The $d = 2$ case is more interesting, leading to one of the most classical structures in computational geometry, the *Voronoi diagram* (see, e.g., the book of [dvOS97]). The idea is to partition the plane into polygonal regions, one per dataset point, each region representing the locus of points that are closer to the respective dataset point than to any other point from the dataset (see Fig. 1-1). For a query q, one just needs to determine the region containing q. This is achieved by performing *point location*, another basic data structure

[1]In literature, the problem is also called "post office problem", "best match file searching problem", "index for similarity search", "vector quantization encoder", "the light-bulb problem", etc.

[2]See Section 2.1 for a definition.

question, which may be solved by, say, persistent binary search trees. This algorithm also achieves $O(n)$ space and $O(\log n)$ query time.

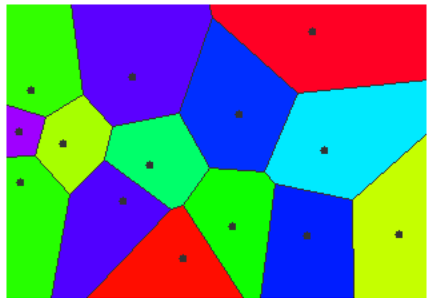


Figure 1-1: A Voronoi diagram of 14 points. Each Voronoi cell is a polygonal region.

How about higher dimensions d ? The generalization of the above algorithm achieves $O(n^{\lceil d/2 \rceil})$ space [Cla88] (see also [Mei93]). Such a space bound is impractical on a dataset of even just a million of points for $d \geq 3$.

There are several practical algorithms known for the case when the dimension d is "low" (say, up to 10 or 20). The first such data structure, called *kd-trees* was introduced in 1975 by Jon Bentley [Ben75], and remains one of the most popular data structures used for searching in multidimensional spaces. Since then, many other multidimensional data structures have been developed, including R-tree, R*-tree, X-tree, SS-tree, SR-tree, VP-tree, metric-trees to name a few; see [Sam06] for an overview.

However, despite decades of intensive effort, the current solutions suffer from either space or query time that is *exponential* in d. In fact, for large enough d, in theory or in practice, they often provide little improvement over a linear time algorithm that compares a query to each point from the database [WSB98]. This phenomenon is often called "the curse of dimensionality".

In recent years, several researchers proposed methods for overcoming the running time bottleneck by using *approximation* (e.g., [AMN$^+$98, Kle97, IM98, KOR00, HP01, KL04, DIIM04, CR04, Pan06, AC06], see also [SDI06, Ind03c]). In that formulation, the algorithm is allowed to return a point whose distance from the query is at most c times the distance from the query to its nearest points; $c > 1$ is called the *approximation factor.*

The appeal of this approach is that, in many cases, an approximate nearest neighbor is almost as good as the exact one. In particular, if the distance measure accurately captures the notion of user quality, then small differences in the distance should not matter. Moreover, an efficient approximation algorithm can be used to solve the *exact* nearest neighbor problem, by enumerating *all* approximate nearest neighbors and choosing the closest point[3]. For many data sets this approach results in very efficient algorithms (see e.g., [ADI$^+$06]).

In this thesis, we propose new algorithms for the approximate NN problem, under the Euclidean and other distances. Before continuing with the presentation, we formalize the definitions of the (approximate) NN problem.

[3]See Section 3.1.1 for more information about exact algorithms.

1.2 Definition of the Nearest Neighbor Problem

We now formalize the problems studied in this thesis. The NN problem is an example of an *optimization* problem: the goal is to find a point which minimizes certain objective function (in this case, the distance to the query point). In contrast, in this thesis we concentrate on the *decision* version of the problem. To simplify the notation, we say that a point p is an R*-near neighbor* of a point q if the distance between p and q is at most R (see Figure 1-2). In this language, an algorithm either returns one of the R-near neighbors, or concludes that no such point exists, for some fixed parameter R.

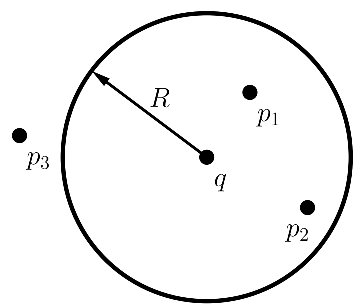


Figure 1-2: An illustration of an R-near neighbor query. The nearest neighbor of the query point q is the point p_1. However, both p_1 and p_2 are R-near neighbors of q.

Naturally, the nearest and near neighbor problems are related. It is easy to see that the nearest neighbor problem also solves the R-near neighbor problem — one can simply check if the returned point is an R-near neighbor of the query point. The reduction in other direction is somewhat more complicated, and involves creating *several* instances of the near neighbor problem, for different values of R. During the query time, the data structures are queried in the increasing order of R. The process is stopped when a data structure reports an answer. See [HP01] for a reduction of this type with theoretical guarantees.

In the rest of this thesis we focus on the approximate near neighbor problem. The formal definition of the approximate version of the near neighbor problem is as follows.

Definition 1.2.1 (Randomized c-approximate R-near neighbor, or (c, R)-NN, or c-NN)**.** *Given a set D of points in a d-dimensional space $\mathbb{R}^d$, and parameters $R > 0$, $\delta > 0$, construct a data structure such that, given any* query *point q, if D contains an R-near neighbor of q, it reports some cR-near neighbor of q in D with probability at least $1 - \delta$.*

For simplicity, we often skip the word "randomized" in the rest of the thesis. In those situations, we will assume that δ is an absolute constant bounded away from 1 (say, 1/2). Note that the probability of success can be amplified by building and querying several instances of the data structure. For example, constructing two independent data structures, each with $\delta = 1/2$, yields a data structure with probability of failure $\delta = 1/2 \cdot 1/2 = 1/4$.

We also define a related *reporting* problem.

Definition 1.2.2 (Randomized R-near neighbor reporting)**.** *Given a set D of points in a d-dimensional space $\mathbb{R}^d$, and parameters $R > 0$, $\delta > 0$, construct a data structure such that, given any query point q, each of the R-near neighbors of q in D is reported with probability at least $1 - \delta$.*

Note that the latter definition does *not* involve an approximation factor. Also, unlike in the case of the approximate near neighbor, here the data structure can return many (or even all) points, if a large fraction of the data points are located close to the query point. As a result, one cannot give an *a priori* bound on the running time of the algorithm. However, as we point out later, the two problems are intimately related. In particular, the algorithms in this thesis can be easily modified to solve both c-NN and the reporting problems. We will expand on this issue for one of our algorithms in Section 3.1.1.

We consider the c-NN problem under other distances, such as the *Hamming distance*. The Hamming distance between two binary strings of length d is the number of differing positions between the two strings. It is also essentially equivalent to the *Manhattan distance*, or ℓ_1, where the distance between two real vectors is the sum of the absolute differences between the coordinates (see Section 2.1 for formal definitions).

1.3 Nearest Neighbor Problem: History and Our Contributions

We now present a history of approaches to the high-dimensional NN problem, highlighting our contributions. These include both improving the old approaches and proposing new ones. Our description is composed of several parts.

I. We start by describing the classical approach to NN under the Euclidean and Hamming distances. These algorithms are based on hashing, and, in particular, on the Locality-Sensitive Hashing scheme. Our contribution is to give a new Locality-Sensitive Hashing algorithm for the Euclidean space, which is *near-optimal* in this class of NN algorithms.

II. We then discuss NN under other, harder distances, such as the standard edit distance on strings. A classic approach to dealing with NN under such distances is to reduce the problem to NN under Hamming distance, where efficient algorithms exist. We show that, while this approach gets us some mileage, it meets a *concrete barrier* at a nearly-logarithmic approximation regime for many such hard distances.

III. This obstacle leads us to the next section, where we suggest a *new approach to NN* that overcomes this barrier. We provide a specific example of a distance, a variant of edit distance, for which we obtain an NN algorithm with approximation exponentially better than what is possible via the aforementioned approaches.

IV. The NN story would be incomplete without the study of NN under the ℓ_∞ distance, which turns out to be the new obstacle in our newly proposed approach. The ℓ_∞ distance between two d-dimensional vectors is the maximum difference among the d coordinates, and it stands alongside with classical distances such as the Hamming (ℓ_1) and Euclidean (ℓ_2) distances. Yet, ℓ_∞ remains intriguingly different from them: it admits an NN solution, with an unusual approximation — double-logarithmic in the dimension d. We prove a data structure lower bound for ℓ_∞, which proves that this approximation is *optimal* in the algorithm's computational model.

V. While the main focus of the present thesis is the NN problem, our main algorithmic techniques naturally *extend to other problems* as well. For the techniques underlying both of our approaches ("old" and "new"), we show how they lead to improved algorithms for other problems, such as the classical problem of estimating the edit distance between two strings, in near-linear time.

1.3.1 Classical approach: Hashing

The algorithms of [KOR00, IM98] were the first to circumvent the curse of dimensionality for the approximate near neighbor problem. For an approximation of $1 + \epsilon$, their data structures support queries in $O(d \log n)$ time (later improved in [AC06]), and use space which is polynomial in n. These algorithms are essentially full-indexing algorithms: they prepare an index table for "each possible" query point (after an appropriate non-trivial discretization of the space), where each cell stores the correct answer. Thus, to answer a query, these algorithms use only a single look-up in a specially-constructed hashing table. Unfortunately, the exponent in the space bounds is roughly $O(1/\epsilon^2)$ (for $\epsilon < 1$), where "big-Oh" constant is "non-negligible". Even for, say, $\epsilon = 1$, the space used by the data structure is large enough so that the algorithm becomes impractical even for relatively small datasets. Furthermore, as we show in this thesis, the space of $n^{\Omega(1/\epsilon^2)}$ is essentially tight when the data structure performs one, or a small number of look-ups[4] into the data-structure.

An alternative approach, introduced in [IM98, GIM99], uses much smaller space while preserving a sub-linear query time. It relies on the concept of *locality-sensitive hashing (LSH)*. The key idea is to hash the points using several hash functions so as to ensure that, for each function, the probability of collision is much higher for points which are close to each other than for those which are far apart. Then, one can determine near neighbors by hashing the query point and retrieving elements stored in buckets containing that point. In [IM98, GIM99] the authors provided such locality-sensitive hash functions for the case when the points live in the Hamming space. The algorithm also extends to the Euclidean space by using embeddings (see definition in Section 1.3.2), however the extension adds additional complexity to the algorithm. A followup work, [DIIM04] introduced LSH functions that work directly in Euclidean space and result in a (slightly) faster running time. The latter algorithm forms the basis of E^2LSH package [AI05] for high-dimensional similarity search, which has been used in many applied scenarios.

The standard LSH-based algorithms have guarantees of the form: n^ρ query time and $n^{1+\rho}$ space, where the parameter $\rho < 1$ depends on the approximation c. For example, [DIIM04] achieve ρ slightly below $1/c$. More recently, [Pan06] proposed a different method of utilizing locality-sensitive hash functions, which results in near-linear space, at the cost of a somewhat higher query time.

The natural question raised by this line of research is: what is the smallest exponent ρ achievable via the locality-sensitive hashing approach? It has been conjectured that one can achieve $\rho \leq 1/c^2$ (see, e.g., [Ind02a]). The conjecture was motivated by the fact that an algorithm with such exponent exists for the closely related problem of finding the *farthest neighbor* [Ind03a].

[4] More formally, cell probes, which can be viewed as adaptive look-ups. See more on the cell probe model in Chapter 8.

Our contributions. We essentially resolve the issue by providing an algorithm with query time $dn^{\rho(c)}$ using space $dn^{1+\rho(c)}$, where ρ becomes arbitrarily close to $1/c^2$:

$$\lim_{n\to\infty} \rho(c) = 1/c^2.$$

This significantly improves over the earlier running time of [DIIM04]. In particular, for $c = 2$, our exponent tends to 0.25, while the exponent in [DIIM04] was around 0.45. Moreover, [MNP06] show that LSH-based algorithms cannot achieve $\rho < 0.462/c^2$. Thus, the exponent of the running time of our algorithm is essentially optimal, up to a small constant factor. We also note another recent related lower bound: [PTW08] show that for any data structure with a constant number of look-ups, the space must be at least $n^{1+\Omega(1/c^2)}$.

Our result also extends to the regime of NN algorithms with a near-linear space. Specifically, we show how to combine our algorithm with the techniques from [Pan06] to obtain an algorithm with $\tilde{O}(dn)$ space and $dn^{O(1/c^2)}$ query time.[5] This improves over [Pan06]'s result, which achieves $dn^{O(1/c)}$ query time.

Our main NN algorithm immediately implies improved algorithms for several other approximate problems in high dimensional spaces. For example, it is known [IM98, Ind01c] that the c-approximate minimum spanning tree (MST) problem for n points in the Euclidean space can be computed by using $O(n \log n)$ calls to the c-approximate near neighbor oracle for that space. Thus, our result implies a $dn^{1+1/c^2+o(1)}$-time algorithm for the c-approximate MST problem. Other problems for which similar improvement is obtained include (dynamic) closest pair and facility location [Ind01c].

We present the state-of-the-art algorithms for the NN problem under the Euclidean distance in Table 1.2. All algorithms presented there are hashing-based algorithms.

Paper	**Type**	**Query time**	**Space**	**Comments**
[KOR00, IM98]	Upper bound	$O(d \log n)$, 1 look-up	$n^{O(1/\epsilon^2)}$	$c = 1 + \epsilon \le 2$
This thesis	Lower bound	$O(1)$ look-ups	$n^{\Omega(1/\epsilon^2)}$	
[IM98, GIM99]	Upper bound	dn^ρ	$n^{1+\rho}$	$\rho = 1/c$
[DIIM04]	Upper bound	dn^ρ	$n^{1+\rho}$	$\rho < 1/c$
This thesis	Upper bound	dn^ρ	$n^{1+\rho}$	$\rho = 1/c^2 + o(1)$
[MNP06]	Lower bound	$\rho \ge 0.45/c^2$ for LSH-based algorithms		
[PTW08]	Lower bound	$O(1)$ look-ups	$n^{1+\Omega(1/c^2)}$	
[Pan06]	Upper bound	dn^ρ	n	$\rho = O(1/c)$
This thesis	Upper bound	dn^ρ	n	$\rho = O(1/c^2)$

Table 1.2: Space and time bounds for hashing-based data structures for NN under the Euclidean distance. All algorithms, except [KOR00], are LSH-based. Factors polynomial in $\log n$ and $1/\epsilon$, as well as an additive term of dn in the space bound, are omitted for clarity.

An external application. We also give an application of the LSH-based techniques to another problem of interest, namely the problem of *approximating kernel spaces.* A kernel space is a high-dimensional Euclidean space that is given only implicitly, via its inner product. For example, the *Gaussian kernel* is the (infinite-dimensional) space of vectors $\phi(x)$, for

[5] Following standard convention, $\tilde{O}(f(n))$ denotes $O(f(n) \cdot \log^{O(1)} f(n))$.

$x \in \mathbb{R}^d$ where $d \in \mathbb{N}$ is fixed, such that $K(x,y) = \phi(x) \cdot \phi(y) = e^{-\|x-y\|^2}$ for all $x, y \in \mathbb{R}^d$. In general, the map $\phi(x)$ necessarily has to be high-dimensional (or even infinite-dimensional) and is thus inefficient to work with. We show how one can efficiently approximate such maps ϕ using LSH techniques. As shown in [RR07], such approximations may lead to significant improvements in classification algorithms, and other related kernel algorithms in machine learning.

1.3.2 Classical approach: Embeddings and sketching

So far we have discussed the Euclidean and Hamming distances only. Not surprisingly, these are just two out of the many distances for which we desire efficient NN algorithms. For example, a natural distance between strings is the classical edit distance, or Levenshtein distance, which is equal to the number of insertions/deletions/substitutions to transform one string into the other [Lev65] (see definitions and motivation for this and other distances in Section 2.1). Most of such distances are no easier than the Euclidean or Hamming distance — in fact, often provably harder — and thus require new NN algorithms.

Methodologically, given a large number of distances, it is desirable to design a common approach to all (or most) of them at the same time. Indeed, in the last decade, researchers identified two such approaches: embeddings and sketching.

Below we describe each of these two approaches. In the following description, we focus on the edit distance as a motivating example of a hard distance. We will elucidate that, while these approaches give many improved algorithms, they suffer from fundamental limitations to achieving very efficient algorithms.

Embeddings

A natural way to deal with the NN problem under a new distance is to reduce it to NN under a distance for which we have an efficient algorithm, such NN under Hamming or Euclidean distances. Embeddings are perhaps the most natural way to achieve this goal.

An *embedding* is a mapping from one metric space (the "guest" space) to another (the "host" space), which preserves the distances between every pair of points, up to a multiplicative factor called *the distortion*. For example, an embedding of edit distance into ℓ_1 is a mapping f from the strings into the real space, such that for all strings x, y,

$$\mathrm{ed}(x,y) \leq \|f(x) - f(y)\|_1 \leq \alpha \cdot \mathrm{ed}(x,y),$$

where $\alpha \geq 1$ is the embedding's distortion (see formal definition in Section 2.2), and $\mathrm{ed}(x,y)$ is the edit distance between x and y. An embedding with distortion α gives an efficient near neighbor data structure for approximation, say, 2α, by applying the embedding and utilizing the NN algorithms from [IM98, KOR00].[6]

The ℓ_1 space in particular has proven to be a remarkably convenient host space. This is because:

- it is *rich* — many interesting and useful metrics can be embedded into it with good distortion (see Table 1.3), and
- it is *tractable* — several computational problems in it, most notably NN, admit efficient algorithms.

[6] One would also require that the embedding $f(x)$ is efficiently computable. Since this requirement is often automatically satisfied, we will mostly ignore this issue.

Besides the NN application, the embeddings approach was successful in a number of other applications including approximation algorithms for the sparsest-cut in a graph [LLR95, AR98, ARV04, ALN05], sketching under edit distance [OR07, CK06] and under Earth-Mover Distance [Cha02, IT03, NS07, AIK08]. See also the surveys of [Ind01a, Lin02, Mat02, IM03].

Naturally, researchers were keen to find the least distortion of embedding into ℓ_1 of various metrics — e.g., the problem of embedding of edit distance is cited in Matoušek's list of open problems [Mat07], as well as in Indyk's survey [Ind01a].

Metric	**Upper bound**	**Lower bound for** ℓ_1	**Lower bound for ultra-sketchables**
Edit distance on $\{0,1\}^d$	$2^{O(\sqrt{\log d \log\log d})}$ [OR07]	$\Omega(\log d)$ [ADG+03, KN06, KR06, AK07]	$\Omega\left(\log d \;/\; \log\log d\right)$ *This thesis*
Ulam distance (edit distance on non-repetitive strings)	$O(\log d)$ [CK06]	$\Omega\left(\log d \;/\; \log\log d\right)$ [Cor03], *This thesis*	$\Omega\left(\log d \;/\; \log\log d\right)$ *This thesis*
Block edit distance, edit distance with moves	$O(\log d \log^* d)$ [CPSV00, MS00, Cor03, CM07]	—	—
EMD over $[d]^2$	$O(\log d)$ [Cha02, IT03]	$\Omega(\sqrt{\log d})$ [NS07]	—
EMD over $\{0,1\}^t$ (for sets of size s)	$O(\log s \log t)$ [AIK08]	$\Omega(\log s)$ [KN06]	$\Omega(\log s)$ *This thesis*

Table 1.3: Distortion for embedding of various metrics into ℓ_1, as well as lower bounds for embedding into "ultra-sketchable spaces", namely spaces admitting constant-sized sketches. EMD refers to the Earth-Mover Distance. See definitions and motivation of the presented distances in Section 2.1.

However, it was recently discovered that this approach has inherent limitations. In particular, for the aforementioned problems, embedding into ℓ_1 cannot result in algorithms with constant approximation factors [KV05, KN06, KR06, DKSV06, NS07, AK07, AIK08, CKN09]. Table 1.3 summarizes known bounds on edit distance, some of its variants, and variants of the Earth-Mover Distance (EMD).

Hence, a natural question arises: are there other host spaces that are *richer*, yet *tractable*? Prior to this thesis, very little was known. For example, the "next natural" candidate for a host space is the squared-ℓ_2 space, the real space with squared Euclidean distance (see formal definition in Section 2.1). This space has an efficient NN solution (which follows directly from NN under the Euclidean distance), and it is provably richer than ℓ_1 in general [KV05, KR06, DKSV06, CKN09]. Yet, even for this space, the only lower bound on embedding edit distance into it was of $3/2$ [ADG+03]. Furthermore, for even more general spaces where we can still apply LSH-based algorithms, such as higher powers of ℓ_2, the bound decays to essentially 1.

Our contributions. In this thesis, we rule out the possibility of low-distortion embedding into a large class of host spaces for the edit distance and its important variant, the Ulam distance. This class of host spaces, termed "ultra-sketchable spaces", includes ℓ_1, any fixed power of ℓ_2, as well as, more generally, all spaces which are amenable to the hashing-based techniques described in Section 1.3.1. We describe this class in more detail in the "Sketching" section below.

In terms of the above discussion, our lower bound implies that both edit distance over $\{0,1\}^d$ and the Ulam distance do not embed into any fixed power of ℓ_2 with distortion better than $\Omega(\log d \ / \ \log\log d)$. We note that in the case of Ulam distance, our result is the first super-constant bound even for embedding into ℓ_1, and, in fact, it nearly matches the upper bound of $O(\log d)$ from [CK06]. See also Table 1.3.

At this moment, it is also worth mentioning that one could also consider ℓ_∞ space as another potential candidate for a host space (described in more detail in Section 1.3.4 below). While the ℓ_∞ space is actually rich enough to contain edit distance, it is not tractable: a low-distortion embedding requires very high dimension, roughly exponential in the strings' length d. This follows from the corresponding lower bound on Hamming distance (see [JL01, Section 8]), and the fact that edit distance is no easier than Hamming distance (see, e.g., [BJKK04], for a reduction).

Sketching

Sketching is similar to the embedding approach and can be seen as an embedding into a host space of a more "computational" flavor. Formally, the sketch of a string x is a (randomized) mapping of x into a short "fingerprint" $f(x)$, such that sketches of two strings, $f(x)$ and $f(y)$, are sufficient to distinguish between the case where edit distance is $\text{ed}(x,y) \le R$, and the case where $\text{ed}(x,y) > \alpha R$, for fixed approximation factor $\alpha > 1$ and parameter $R > 1$, with good probability (say, at least 2/3). The main parameter of a sketching algorithm is its *sketch size*, the length of $f(x)$ (measured in bits).[7] Of particular interest is the case of constant-size sketches, as we show next; we refer to the spaces admitting constant-sized sketches as *ultra-sketchable*.

Beyond the syntactic similarity to an embedding, ultra-sketchable spaces are in fact a generalization of the embeddings approach presented above, albeit with an (arbitrarily small) constant factor loss in the approximation factor. This follows from the following two facts:

$\ell_1 \implies$ ultra-sketchable: Embedding into ℓ_1, or even powers of ℓ_2, implies a sketch of small size.

Namely, [KOR00] show a sketching algorithm for the ℓ_1 distance achieving $1+\epsilon$ approximation in $O(1/\epsilon^2)$ space, for any $\epsilon > 0$. Hence, any space embeddable into ℓ_1 (or powers of ℓ_2)[8] also admits a constant-sized sketch, with only a $1+\epsilon$ loss in approximation.

ultra-sketchable $\implies$ NN: Existence of small sketches implies efficient NN algorithms.

In particular, if a space admits a sketch of size s, for some approximation α, then one immediately obtains an NN algorithm for the same approximation α, with $n^{O(s)}$ space and query time essentially proportional to the time to compute the sketch. This algorithm is essentially a full-indexing algorithm, which stores an answer for all possible sketches of the query point. Other space/query-time trade-offs are also possible [AK08a].

[7] The sketching model is also equivalent to a (randomized) simultaneous communication protocol; see also Chapter 7.

[8] For a formal argument in the case of powers of ℓ_2, see the argument in Corollary 7.0.7.

Indeed, the NN data structure under the Hamming distance of [KOR00] can be viewed as an instantiation of this approach, resulting in an NN data structure with $n^{O(1/\epsilon^2)}$ space and $O(d \log n)$ query time.[9]

The sketching model is also important as a basic computational notion for massive data sets [FM85, AMS99, BBD$^+$02], and in recent years, an intensive research effort has led to a number of sketching algorithms [Mut03, Mut09]. For instance, sketching can be useful for quickly estimating the distance (e.g., as a filtering step to speed up the linear-scan NN algorithm).

Prior to this thesis, the only sketching lower bounds were known for the ℓ_p spaces, for $p \in [1, \infty]$ [Woo04, SS02a, BJKS04]. With regards to the edit distance, in 2004, [BJKK04] wrote that "The state of affairs indicates that proving sketching lower bounds for edit distance may be quite hard."

Our contributions. In this thesis, we prove a sketching lower bound for edit distance over $\{0,1\}^d$ and Ulam distance, as well as the Earth-Mover Distance over $\{0,1\}^d$. For edit and Ulam distances, we show that constant-sized sketches cannot achieve an approximation below $\Omega(\log d \ / \ \log\log d)$. The lower bound implies the aforementioned non-embeddability results, by the "$\ell_1 \Longrightarrow$ ultra-sketchable" implication from above. In a very recent manuscript [AJP10], we show how to combine this lower bound with the information complexity tools of [CSWY01, BJKS04] in order to extend the lower bound to sketches of nearly-logarithmic size. For Ulam distance, this bound matches the upper bound, up to a constant in the exponent (see the next section).

We note that our sketching lower bound is also the first lower bound of *computational* flavor for the edit distances, thus providing rigorous evidence for the perceived hardness of the edit distance (see also the discussion at the beginning of Chapter 7, as well as in Section 7.7).

1.3.3 New approach: Iterated product spaces

In the previous section, we have unveiled a barrier for the classical approaches to NN for some "hard" distances of interest, such as the edit distance and its variant, the Ulam distance. Namely, we have shown that, for any approach based on reducing edit distance to classical host spaces — such as ℓ_1, Euclidean space, squared-ℓ_2, ℓ_∞, or constant-size sketches — we will not be able to go below near-logarithmic approximation factor. Thus, the following question re-emerges: can we identify new, richer host spaces, which, at the same time, admit efficient NN algorithms?

Our contributions. In this thesis we answer positively to this question. We propose an alternative way to achieve generic NN algorithms, and provide a concrete implementation for a metric as a proof-of-concept. We mostly focus on the Ulam distance, the edit distance on non-repetitive strings. For this distance, our approach leads to an efficient NN algorithm with approximation that is nearly double-logarithmic in the strings' length, which is an *exponential* improvement over what is possible via the previously mentioned approaches, such as embedding into ℓ_1 (see Table 1.3).

[9] We note that the algorithm presented in [KOR00] is a bit more complicated than that because they manage to obtain a certain deterministic solution to the NN problem. We will not discuss this version in the present thesis.

Our approach is to consider a class of somewhat unusual host spaces, namely the *iterated products* of standard spaces like ℓ_1 and low-dimensional ℓ_∞. We show that these spaces exhibit a better balance between the desired *richness* and *tractability*. Indeed, we identify a sweet spot: the spaces are rich enough to accommodate the intended guest spaces with only a constant distortion, while admitting quite efficient algorithms.

An example of our host spaces is the space

$$\bigoplus_{(\ell_2)^2}^{k} \bigoplus_{\ell_\infty}^{l} \ell_1^m$$

which is a combination of the standard ℓ_1, ℓ_2, and ℓ_∞ norms. The symbol $\bigoplus_{\ell_\infty}^{l}$ is an ℓ_∞-*product* operator that takes the ℓ_1^m space and produces a new space, $\bigoplus_{\ell_\infty}^{l} \ell_1^m$, with the following distance. Imagine the two points as two-dimensional matrices, of size $l \times m$, and compute the difference matrix. On each row, apply the ℓ_1 norm, reducing the matrix to a vector. Then, on the resulting l-dimensional vector, apply the ℓ_∞ norm, yielding the $\bigoplus_{\ell_\infty}^{l} \ell_1^m$ distance. Finally, we iterate this operation again, on three-dimensional arrays, with squared-ℓ_2 on the outside, thus obtaining the intended distance. Formally, the space contains points $x \in \mathbb{R}^{k \cdot l \cdot m}$ under the following distance function[10]

$$\mathsf{d}_{2^2,\infty,1}(x,y) = \sum_{a=1}^{k} \Big(\max_{b=1,2,\dots,l} \Big\{ \sum_{c=1}^{m} |x_{a,b,c} - y_{a,b,c}| \Big\} \Big)^2,$$

where $x_{a,b,c}$ stands for coordinate $(a-1)lm + (b-1)m + c$ of x.

For the Ulam distance specifically, we show that we can embed it into the above host space with only a *constant* distortion. Furthermore, we show that the above host space admits an efficient NN solution with a nearly double-logarithmic approximation.

Besides the Ulam distance, we also consider the Earth-Mover Distance (EMD), which is a metric of interest for computing similarity between two images (see Section 2.1 for definition and motivation). For EMD, our partial results indicate that this new approach may be applicable here too. Like for Ulam distance, the embedding of EMD into ℓ_1 and related spaces such as squared-ℓ_2 provably requires high distortion.

In fact, we show how to construct an efficient NN algorithm for *any* iterated product space of any ℓ_p's, with an approximation polynomial in a double-logarithmic factor. In particular, we obtain *the first* NN algorithm under the standard ℓ_p norm for $p \in (2, \infty)$; no algorithms for this norm were known previously.

External applications. Finally, we show that embeddings into products spaces may be helpful for applications other than NN problem as well.

First, we show how to estimate the (standard) edit distance between two strings in near-linear time, with a greatly improved approximation factor. While this classic problem has a textbook dynamic programming solution, the algorithm runs in quadratic time. This runtime has not been improved since 1980, and even then the improvement was only by a poly-logarithmic factor [MP80]. Years of research on *approximate* algorithms yielded several near-linear time algorithms with approximation of the order d^c for different constants c, where d is the length of the strings (state-of-the-art was $c = 1/3$ [BES06]). We achieve approximation factor of $2^{\tilde{O}(\sqrt{\log d})}$, which is smaller than d^ϵ for any small constant $\epsilon > 0$.

[10] We note that, formally, this function does not satisfy the triangle inequality; however, this aspect will be of little importance here.

Second, we provide a algorithm for sketching of the Ulam distance. In fact, we design a *streaming* algorithm for estimating the Ulam distance between two strings in polylogarithmic space. Our algorithm also answers an open question of [AJKS02], who consider the same problem for the Kendall-tau (number of inversions) distance.

We summarize the algorithms obtained using product spaces techniques in Table 1.4.

Problem	**Reference**	**Approx.**	**Comments**
Nearest Neighbor[a] under Ulam distance	[CK06]	$O(\log d)$	
	[Ind04]	$3^{\alpha-1}$	space $n^{O(d^{1/\alpha})}$
	This thesis	$\Omega\left(\frac{\log d}{\log\log d}\right)$	for embedding in ℓ_1, ultra-sketchables
	This thesis	$O(\log\log d)$	$d^{O(1)}n^{\epsilon}$ query time
Sketching Ulam distance (streamable)	[CK06]	$O(\log d)$	$O(1)$ size sketch
	This thesis, [AJP10]	$O(1)$	$\Omega(\log d\ /\ \log\log d)$ size
	This thesis	$O(1)$	$(\log d)^{O(1)}$ size
Computing edit distance on $\{0,1\}^d$	[BES06]	$d^{1/3+o(1)}$	in $d^{1+o(1)}$ time
	This thesis	$2^{\tilde{O}(\sqrt{\log d})}$	in $d^{1+o(1)}$ time
Nearest Neighbor under EMD over $[d]^2$	[Cha02, IT03]	$O(\log d)$	
	[NS07]	$\Omega(\sqrt{\log d})$	for embedding into ℓ_1
	[ADIW09]	$O(\alpha)$	$n^{d^{1/\alpha}}$ space, $(d\log n)^{O(1)}$ query time
	This thesis	$O(\alpha\log\log n)$	$2^{d^{1/\alpha}}n^{1+\epsilon}$ space, $d^{O(1)}n^{\epsilon}$ query time
Nearest Neighbor under ℓ_p for $p>2$	*This thesis*	$O(\frac{1}{\epsilon}\log\log d)$	$O(dn^{1+\epsilon})$ space, $\tilde{O}(dn^{\epsilon^p})$ query time

[a] Unless mentioned otherwise, query time is $d\log^{O(1)} n$ and space is $(dn)^{O(1)}$.

Table 1.4: Results based on product spaces techniques for the edit over $\{0,1\}^d$, Ulam, and EMD metrics, as compared with the previous bounds. $\epsilon>0$ and $\alpha\in\mathbb{N}$ are arbitrary.

1.3.4 Odd man out: The ℓ_∞ distance

The ℓ_∞ distance is another classical norm, next to Hamming (ℓ_1) and Euclidean distance (ℓ_2). However ℓ_∞ seems to be intriguingly different from these other norms, and remains much less understood.

In fact, there was precisely one worst-case[11] result on NN under the ℓ_∞ norm. [Ind98] achieves an NN algorithm for d-dimensional ℓ_∞ with $O(\log\log d)$ approximation, which requires space $dn^{1+\epsilon}\log^{O(1)} n$ and $d\cdot\log^{O(1)} n$ query time, for any fixed $\epsilon>0$.

The ℓ_∞ space is of interest as a potential host space for some of the "hard" metrics. The motivation comes from a result of Matoušek that states that *any* metric on n points can be embedded into ℓ_∞ of dimension $d=O(cn^{1/c}\log n)$ with $2c-1$ distortion [Mat96], for any $c\in\mathbb{N}$. While this general guarantee on the dimension is too high for many applications, it suggests that ℓ_∞ is a very good target space for trying to embed particular metrics more efficiently. (See also more motivation for ℓ_∞ distance in Chapter 9.)

[11] Heuristic methods have also been devised. On the theoretical side, [AHL01] analyze a brute-force algorithm for the uniform input distribution, showing a bound of $\Theta(nd/\lg n)$.

As already mentioned in the previous section, ℓ_∞ plays a role in embeddings into (iterated) product spaces. For ℓ_∞-product, Indyk [Ind02b] has extended his original NN algorithm from ℓ_∞ to ℓ_∞-product spaces. Using this algorithm for ℓ_∞-product spaces, Indyk [Ind02b] obtained an NN algorithm for the Frechet metric. Even more pertinent to our discussion is the fact that ℓ_∞ distance plays an important role for iterated product spaces, and in particular for the Ulam distance.

In fact, the bottleneck in some of the currently best algorithms for Ulam and Frechet metrics is the ℓ_∞ norm. In particular, one obtains the same unusual double-logarithmic approximation in the polynomial space regime.

Hence the following question emerges: can one improve Indyk's algorithm [Ind98] or prove it is optimal?

Our contributions. In this thesis, we give an indication that Indyk's unconventional double-logarithmic approximation bound may in fact be optimal. Specifically, we prove a lower bound for NN under ℓ_∞, showing that the space/approximation trade-off from [Ind98] is optimal for decision trees and for data structures with constant cell-probe complexity.

1.4 Summary and Organization

We present the detailed exposition of our contributions in two parts. In the first part, we describe the algorithmic results of this thesis, which are summarized as follows:

- We design a new algorithm for NN under the Euclidean space, achieving dn^ρ query time and $dn^{1+\rho}$ space/preprocessing for $\rho = 1/c^2 + o(1)$. We also give a practical variant of the algorithm, based on the Leech lattice. Furthermore, we show how to modify our algorithm to obtain near-linear space and $dn^{O(1/c^2)}$ query time. These results, along with a description of the main technique, Locality-Sensitive Hashing, appear in Chapter 3.

- We give an application of the Locality-Sensitive Hashing to approximating kernel spaces. Our results apply to kernel spaces such as the Laplacian, Gaussian, Jaccard, and Geodesic kernels. This result appears in Chapter 4.

- We present a new approach to NN under "hard" metrics, based on embeddings into iterated product spaces. We design new NN algorithms for a large class of iterated product spaces. As a particular application, we show how this approach yields a new NN algorithm for the Ulam distance, with only a nearly double-logarithmic approximation. These results appear in Chapter 5.

- We present two applications of product spaces to other problems, which extend beyond the NN application. In the first application, we give an algorithm for computing the edit distance between two strings of length d in near-linear time, with $2^{\tilde{O}(\sqrt{\log d})}$ approximation.

  In the second application, we design a sketching algorithm for the Ulam distance, which achieves constant approximation with a sketch of only polylogarithmic size. These results appear in Chapter 6.

In the second part, we present our impossibility results, which are summarized as follows:

- We prove sketching lower bounds for the edit, Ulam, and EMD distances. For edit and Ulam distances, we show that protocols with $O(1)$ bits of communication can only obtain approximation $c \geq \Omega(\log d / \log\log d)$, where d is the length of the input strings. These lower bounds also immediately imply distortion lower bounds for embedding the metrics into ℓ_1 and powers of ℓ_2. These results appear in Chapter 7.

- We prove a lower bound for NN under Hamming and Euclidean distances for $c = 1+\epsilon$ approximation and small query time. Namely, we show that any data structure for the $(1+\epsilon)$-NN, which uses constant number of probes to answer each query, must use $n^{\Omega(1/\epsilon^2)}$ space. This result appears in Chapter 8.

- We prove a lower bound for NN under the ℓ_∞ norm, which matches the upper bound from [Ind98] in the decision tree model. Namely, we show that, for any $\rho > 1$, any decision tree for the $O(\log_\rho \log d)$-NN, with sub-linear depth, must use $n^{\Omega(\rho)}$ space. This result appears in Chapter 9.

We conclude with some open questions raised by our work in Chapter 10.

Parts of this thesis have been published as the following papers and manuscripts: [AI06b, AI08b, ADI+06, AI08a, AIK08, AIK09, AO09, AK07, AJP10, AIP06, ACP08]

Chapter 2

Preliminaries

We now establish our notation and give motivation for some of the studied metrics, such as edit, Ulam, and EMD distances.

2.1 A Primer on Metrics

2.1.1 Edit and Ulam distances

Consider two strings of length d from some alphabet Σ. The *edit distance* (or *Levenshtein distance*) between two strings is the number of insertions, deletions, and substitutions needed to transform one string into the other [Lev65]. We denote the distance between two strings $x, y \in \Sigma^d$ by $\mathrm{ed}(x, y)$.

The edit distance is of fundamental importance in several fields such as computational biology and text processing/searching, and consequently, problems involving edit distance were studied extensively (see [Nav01], [Gus97], and references therein). In computational biology, for instance, edit distance and its slight variants are the most elementary measures of dissimilarity for genomic data, and thus improvements on edit distance algorithms have the potential of major impact.

There are two important variants of edit distance obtained by restricting the set of vectors $x \in \Sigma^d$:

- *standard edit distance metric*, i.e., edit distance on $\{0,1\}^d$,
- *Ulam distance* (or *Ulam metric*), where each vector x is non-repetitive, i.e., each symbol $s \in \Sigma$ appears at most once in x (in this case, we must have $|\Sigma| \geq d$).[1]

There are several motivations for studying the Ulam distance. First of all, Ulam metric models edit distance between strings with limited or no repetitions which appear in several important contexts, most notably in ranking of objects such as webpages (see, e.g., [AJKS02] and [Mar95]). In fact, Ulam distance is one of the standard ways to measure distance between two permutations [Dia88], and a classical motivating example is the following: given a set of books on a shelf, how many operations does one need to perform in order to sort the books? Hence, Ulam distance received a fair amount of attention; see, for example, [AD99] for a treatise of the Ulam distance between two random permutations.

[1] We note that the standard definition for Ulam distance is the number of characters moves to transform x into y, for alphabet $\Sigma = [d]$. However, we will ignore this difference since, up to a factor of two, the two definitions are equivalent.

On a broader scale, we believe Ulam metric presents a concrete milestone towards the seemingly elusive goal of designing algorithms for the standard edit distance (over binary strings). Indeed, there are two reasons for this belief. First, Ulam metric appears to retain one of the core difficulties of the edit distance on general strings, namely the existence of "misalignments" between the two strings. In fact, there is *no known lower bound* that would strictly *separate* Ulam metric from the general edit distance: all known lower bounds are nearly the same (quantitatively) for both metrics. These include non-embeddability into normed spaces results [KR06, AK07], lower bounds on sketching complexity [AK07] (see also Chapter 7), and sub-linear time algorithms [BEK+03]. Second, Ulam distance is no harder than edit distance over binary strings, at least up to a constant approximation, as formalized in the lemma below. Thus, to obtain improved algorithm for the standard edit distance, we must first obtain better algorithms for the Ulam distance.

Lemma 2.1.1. *Let $P, Q \in \Sigma^d$ be two permutations, and let $\pi : \Sigma \mapsto \{0,1\}$ be a random function (i.e., π substitutes every alphabet symbol with 0 or 1 at random). Then*

- $\mathrm{ed}(\pi(P), \pi(Q)) \leq \mathrm{ed}(P, Q)$ *for any choice of* π*, and*
- $\Pr_\pi\left[\mathrm{ed}(\pi(P), \pi(Q)) \geq \Omega(1) \cdot \mathrm{ed}(P, Q)\right] \geq 1 - 2^{-\Omega(\mathrm{ed}(P,Q))}$.

The proof is somewhat involved and appears in Appendix A.

We note that allowing alphabets Σ bigger than $[d]$ does not make Ulam metric harder (in all contexts considered in the thesis). Thus, we will consider that $\Sigma = [d]$ in this thesis. For concreteness, we demonstrate the following reduction from a big alphabet to a smaller alphabet.

Fact 2.1.2. *For any string length d, and alphabet Σ, $|\Sigma| \geq d$, there is a function $f : \Sigma^d \to \Sigma^{|\Sigma|}$ such that for every pair of non-repetitive strings $x, y \in \Sigma^d$, we have that $f(x), f(y)$ are non-repetitive over Σ and*

$$\mathrm{ed}(x, y) \leq \mathrm{ed}(f(x), f(y)) \leq 3\,\mathrm{ed}(x, y).$$

Proof. For given $x \in \Sigma^d$, construct $f(x) \in \Sigma^{|\Sigma|}$ by appending all the alphabet symbols that are missing from x in an increasing order. Then, clearly $\mathrm{ed}(f(x), f(y)) \geq \mathrm{ed}(x, y)$. Furthermore, we claim that $\mathrm{ed}(f(x), f(y)) \leq 3\,\mathrm{ed}(x, y)$. Indeed, edit distance between the starting block of length d of $f(x)$ and of $f(y)$ is $\mathrm{ed}(x, y)$. Also, if $z \leq \mathrm{ed}(x, y)$ is the number of symbols that appear in x but not in y and vice-versa, then the edit distance between the ending block of length $|\Sigma| - d$ of $f(x)$ and $f(y)$ is $2z$. Total edit distance between $f(x)$ and $f(y)$ is at most $3\,\mathrm{ed}(x, y)$. □

There are also a number of other variants of edit distances, such as block edit distance or edit distance with moves; for more details, we refer to [Sah08].

2.1.2 Earth-Mover Distance

The *Earth Mover Distance (EMD)* between two sets of points in the grid $[d]^t$ of equal sizes is defined to be the cost of the minimum cost bipartite matching between the two pointsets.[2] Namely for two sets $A, B \subset [d]^t$, we have

$$\mathrm{EMD}(A, B) = \min_{\pi : A \to B} \sum_{x \in A} \|x - \pi(x)\|_1$$

[2]This distance is also called the *Wasserstein distance*, or the *transportation distance*.

where π ranges over all bijections from A to B.

EMD is a natural metric for comparing sets of geometric features of objects, and, as such, has found applications in visual search and recognition. For example, an image can be represented as a set of pixels in a color space, in which case we have $t = 3$ for a red-green-blue representation. Computing EMD between such sets yields an accurate measure of dissimilarity between color characteristics of the images [RTG00]. In an analogous manner, an image can be represented as a set of representative geometric features, such as object contours [GD04] and other features [GD05a]. This approach to measure dissimilarity of images has lead to some of the state-of-the-art algorithms for image recognition on big collections of images [GD05b, LSP06, GD06].

2.1.3 Normed spaces

The classical ℓ_p distance, for $p \geq 1$, is defined for two points $x, y \in \mathbb{R}^d$ as

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

We will also refer to the distance as ℓ_p^d to indicate that the points come from a d-dimensional space.

The most classical setting is $p = 2$, for which we recover the Euclidean distance $\|x-y\|_2 = \|x - y\|$. Another two important cases are $p = 1$ and $p = \infty$. When $p = 1$, the distance becomes $\|x - y\|_1 = \sum_{i=1}^{d} |x_i - y_i|$, and it is also called the *Manhattan distance*. It is essentially equivalent to the Hamming distance, which we denote by $\mathrm{H}(x, y)$. When $p = \infty$, the distance becomes $\|x - y\|_\infty = \max_{i=1}^{d} |x_i - y_i|$.

We also define powers of the ℓ_p spaces. The most notable example is the squared-ℓ_2 space (also denoted $(\ell_2)^2$). This is a real space $\mathbb{R}^d$ with squared Euclidean distance, $\|x - y\|_2^2$. As such, squared-ℓ_2 is not a metric because it does not satisfy the triangle inequality. It will be useful for us nonetheless. We note that if a set of points S of squared-ℓ_2 satisfies the triangle inequality, then S is called a *negative type metric*.

2.1.4 Product spaces

Product spaces are generalizations of the ℓ_p norms. We define the *ℓ_p-product of a metric* (X, d_X) as follows. Let $k \geq 1$ be an integer. The ℓ_p-product metric is the metric $(X^k, \mathsf{d}_{p,X})$, denoted $\ell_p(X)$ or $\bigoplus_{\ell_p}^k X$, where the distance between $x = (x_1, \ldots x_k) \in X^k$ and $y = (y_1, \ldots y_k) \in X^k$ is

$$\mathsf{d}_{p,X}(x, y) = \left(\sum_{i \in [k]} (\mathsf{d}_X(x_i, y_i))^p \right)^{1/p}.$$

We illustrate the definition with the metric obtained by taking ℓ_∞-product of $X = \ell_1^d$, denoted $\bigoplus_{\ell_\infty}^k \ell_1$. In this space, the points can be thought of as two-dimensional arrays of size $k \times d$ of reals. To compute the norm of a point, we first compute the ℓ_1 norm of each row, thus reducing the array to a vector. Then we apply the ℓ_∞ norm on the resulting vector to obtain the final norm. The distance between two points is the norm of their difference.

In the particular case of $p = 1$, we refer to the ℓ_1-product metric as a *sum-product*. Similarly, we refer to the ℓ_∞-product metric as a *max-product*.

Analogously, a *min-product of* X, denoted $\bigoplus_{\min}^k X$, is a space over X^k, where the "distance" between two points $x = (x_1, \dots x_k)$ and $y = (y_1, \dots y_k)$ is defined as

$$\mathsf{d}_{\min,X}(x, y) = \min_{i \in [k]} \left\{ d_X(x_i, y_i) \right\}.$$

For the distance function in an arbitrary iterated product space, we use the subscript to identify which operations are made in which order. For example $d_{1,X}$ is the distance function of the space $\bigoplus_{\ell_1} X$.

We will also consider products under powers of an ℓ_p norm. For example, $(\ell_2)^2$-product of X is the space X^k under the distance $\mathsf{d}_{2^2,X}(x, y) = \sum_{i=1}^k (\mathsf{d}_X(x, y))^2$. (We denote by $\mathsf{d}_{p^q,\mathcal{M}}$ the distance function of some $\bigoplus_{(\ell_p)^q} \mathcal{M}$.) Abusing terminology, we shall sometimes continue to call this space a metric even when it is not guaranteed to satisfy the triangle inequality.

We note that product spaces, including iterated ones, are examined in the study of the geometry of Banach spaces, see e.g. [JL01, Chapter 1].

2.2 Metric Notions and Embeddings

There are several very close terms for a metric — distance, norm, metric, and space — that we use in this thesis. We now explain our use of these terms.

A space is a tuple $(\mathcal{M}, d_{\mathcal{M}})$ where $\mathcal{M}$ is a set, and $d_{\mathcal{M}} : \mathcal{M}^2 \to \mathbb{R}^+$ is non-negative and symmetric ($d_{\mathcal{M}}(x, y) = d_{\mathcal{M}}(y, x)$ for all $x, y \in \mathcal{M}$). We also call $\mathcal{M}$ *semi-metric* and $d_{\mathcal{M}}$ *semi-distance*. For example, squared-ℓ_2 is a semi-metric.

A metric is a space $(\mathcal{M}, \mathsf{d}_{\mathcal{M}})$ where $\mathsf{d}_{\mathcal{M}} : \mathcal{M}^2 \to \mathbb{R}^+$ is non-negative and symmetric, reflexive ($\mathsf{d}_{\mathcal{M}}(x, y) = 0$ is equivalent to $x = y$), and satisfies the triangle inequality ($\mathsf{d}_{\mathcal{M}}(x, z) \le \mathsf{d}_{\mathcal{M}}(x, y) + \mathsf{d}_{\mathcal{M}}(y, z)$ for all $x, y, z \in \mathcal{M}$). We call $\mathsf{d}_{\mathcal{M}}$ a *distance function*, and, abusing notation, we will call $(\mathcal{M}, \mathsf{d}_{\mathcal{M}})$ a distance as well. For example, ℓ_p's and edit distance are metrics, whereas squared-ℓ_2 is not.

Finally, *a norm* is a tuple $(\mathcal{M}, \|\cdot\|)$, where $\mathcal{M}$ is a vector space and the function $\|\cdot\| : \mathcal{M} \to \mathbb{R}^+$ is positive scalable ($\|ax\| = |a| \cdot \|x\|$ for all scalars a and vectors $x \in \mathcal{M}$), reflexive ($\|x\| = 0$ is equivalent to $x = 0$), and satisfies the triangle inequality ($\|x + y\| \le \|x\| + \|y\|$). The norm naturally gives rise to a metric, given by the distance function $\mathsf{d}_{\mathcal{M}}(x, y) = \|x - y\|$. Abusing notation, we call the resulting metric a norm as well. For example, all ℓ_p's and EMD are norms, whereas edit distance is not a norm.

For two metrics $(\mathcal{M}, \mathsf{d}_{\mathcal{M}})$ and (X, ρ), an *embedding* is a map $\phi : \mathcal{M} \to X$ such that, for all $x, y \in \mathcal{M}$, we have

$$\mathsf{d}_{\mathcal{M}}(x, y) \le \rho(\phi(x), \phi(y)) \le \gamma \cdot \mathsf{d}_{\mathcal{M}}(x, y),$$

where $\gamma \ge 1$ is the *distortion* of the embedding. In particular, we consider all embeddings in this thesis to be non-contracting.

At times we consider embeddings into spaces (as opposed to metrics), which are defined equivalently. For spaces that are powers of a metric, there is also an alternative view on such as embedding. For example, a γ-distortion embedding of some metric $(\mathcal{M}, \mathsf{d}_{\mathcal{M}})$ into squared-ℓ_2 is equivalent to embedding the root of the metric, $(\mathcal{M}, \sqrt{\mathsf{d}_{\mathcal{M}}})$, into ℓ_2, with distortion $\sqrt{\gamma}$.

We say embedding ϕ is *oblivious* if it is randomized and, for any subset $S \subset \mathcal{M}$ of size

n, the distortion guarantee holds for all pairs $x, y \in S$ with good probability (for example, at least $1 - n^{-2}$). The embedding ϕ is *non-oblivious* if it holds for a specific set S (i.e., ϕ is allowed to depend on S).

We also define *γ-near metric* to be a semimetric $(\mathcal{M}, \mathsf{d}_{\mathcal{M}})$ such that there exists some metric $(\mathcal{M}, \mathsf{d}^*_{\mathcal{M}})$ with the property that, for any $x, y \in \mathcal{M}$, we have that $\mathsf{d}^*_{\mathcal{M}}(x, y) \leq \mathsf{d}_{\mathcal{M}}(x, y) \leq \gamma \cdot \mathsf{d}^*_{\mathcal{M}}(x, y)$.

2.3 High-Dimensional Euclidean Space

We describe some facts about the high-dimensional Euclidean space, including the dimensionality reduction method in the Euclidean space.

The geometry of a sphere. For a point $p \in \mathbb{R}^d$, we denote by $B(p, r)$ the ball centered at p with radius r, and we call $\bar{B}(p, r)$ its surface. For a ball with radius r in $\mathbb{R}^d$, we call its surface area $\mathrm{Sur}^d(r)$ and its volume $\mathrm{Vol}^d(r)$. We note that $\mathrm{Sur}^d = S_d \cdot r^{d-1}$ and $\mathrm{Vol}^d(r) = \frac{S_d \cdot r^d}{d}$, where $S_d = \frac{2\pi^{d/2}}{\Gamma(d/2)}$ is the surface area of a ball of radius one (see, for example, [Pis89], page 11).

We will also need a (standard) bound on the volume of a cap of a ball $B(p, r)$. Let $C(u, r)$ be the volume of the cap at distance u from the center of the ball. Alternatively, $C(u, r)$ is the half of the volume of the intersection of two balls of radius r with centers at distance $2u$. Furthermore, let $I(u, r) = \frac{C(u,r)}{\mathrm{Vol}^d(r)}$ be the cap volume relative to the volume of the entire sphere. We can bound $I(u, r)$ as follows.

Fact 2.3.1. *For any $d \geq 2$ and $0 \leq u < r$,*

$$\frac{A_l}{\sqrt{d}} \left(1 - \left(\tfrac{u}{r}\right)^2\right)^{d/2} \leq I(u, r) \leq \left(1 - \left(\tfrac{u}{r}\right)^2\right)^{d/2}.$$

In particular, we have

$$\frac{A_l}{\sqrt{d}} \exp\left[-\frac{d}{2} \cdot \frac{(u/r)^2}{1-(u/r)^2}\right] \leq I(u, r) \leq \exp\left[-\frac{d}{2} \cdot (u/r)^2\right].$$

Proof. The result follows immediately from Lemma 9 of [FS02], which gives bounds on the ratio of the surface areas of the cap to that of the ball. Specifically, note that $I(u, r)$ has the following form

$$I(u, r) = \frac{C(u,r)}{\mathrm{Vol}^d(r)} = \int_u^r \frac{S_{d-1}}{d-1}(r^2 - y^2)^{\frac{d-1}{2}} dy \cdot \left(\frac{S_d}{d} r^d\right)^{-1} = \frac{d}{d-1} \cdot \left(\int_u^r S_{d-1}(r^2 - y^2)^{\frac{d-1}{2}} dy \cdot \left(S_d r^d\right)^{-1}\right).$$

The quantity $\int_u^r S_{d-1}(r^2 - y^2)^{\frac{d-1}{2}} dy \cdot \left(S_d r^d\right)^{-1}$ represents precisely the ratio of the surface area of the cap $C(u, r)$ (excluding the base) to the surface area of a ball of radius r in the $(d + 1)$-dimensional space. This ratio is bounded [FS02] as

$$\frac{A_l}{\sqrt{d+1}} \left(1 - \left(\tfrac{u}{r}\right)^2\right)^{\frac{d}{2}} \leq \int_u^r S_{d-1}(r^2 - y^2)^{\frac{d-1}{2}} dy \cdot \left(S_d r^t\right)^{-1} \leq \frac{1}{2} \left(1 - \left(\tfrac{u}{r}\right)^2\right)^{\frac{d}{2}}.$$

Thus, multiplying the above bounds by $\frac{d}{d-1}$, we obtain that

$$\frac{d}{d-1} \cdot \frac{A_l}{\sqrt{d+1}} \left(1 - \left(\frac{u}{r}\right)^2\right)^{d/2} \le I(u,r) \le \frac{d}{d-1} \cdot \frac{1}{2} \left(1 - \left(\frac{u}{r}\right)^2\right)^{d/2}.$$

Using the standard estimate that $\exp\left[-\frac{x}{1-x}\right] \le 1 - x \le \exp[-x]$ for $x \in (0,1)$, we obtain the conclusion. □

Dimensionality reduction. We will also use the Johnson-Lindenstrauss lemma on the (random) dimensionality reduction in the Euclidean spaces [JL84]. Informally, the Johnson-Lindenstrauss lemma states that, for a set of n points in a high-dimensional space, a projection to a random subspace of dimension only $t = O(\frac{\log n}{\epsilon^2})$ preserves all the inter-point distances up to a $1 \pm \epsilon$ factor. In particular, we use the following form of the lemma (see [JL84, IM98, DG99]). Consider a random matrix $A \in M_{t,d}$ by choosing each element of A from normal distribution $N(0,1)$, multiplied by a scaling factor $\frac{1}{\sqrt{t}}$. This matrix A represents the random linear projection from $\mathbb{R}^d$ to $\mathbb{R}^t$.

Fact 2.3.2 ([JL84, IM98, DG99])**.** *For any vector $v \in \mathbb{R}^d$ and any constant $\epsilon \in (0,1)$, we have*

$$\Pr_A \left[\left| \|Av\| - \|v\| \right| > \epsilon \|v\| \right] \le O(t) \cdot \exp\left[-t\epsilon^2/12\right].$$

Fact 2.3.3 ([IM98])**.** *For any vector $v \in \mathbb{R}^d$ and any constant $\alpha \ge 2$, we have*

$$\Pr_A[\|Av\| > \alpha\|v\|] \le \exp\left[-\Omega(t\sqrt{\alpha})\right].$$

2.4 Probability

For a random variable x from some domain $\mathcal{D}$, we denote its probability distribution functions (pdf) by P_x. We use the notation $x \leftarrow \mathrm{P}_x$ to say that x is drawn from the distribution P_x.

For two random variables x, y with pdfs P_x and P_y, the *statistical distance* (or, *total variation distance*) between x and y is defined as the half of the L_1 distance between P_x and P_y:

$$\tfrac{1}{2}\|\mathrm{P}_x - \mathrm{P}_y\|_1 = \frac{1}{2}\int_{z \in \mathcal{D}} |\mathrm{P}_x(z) - \mathrm{P}_y(z)| dz.$$

We denote this distance by $\mathrm{TV}(\mathrm{P}_x, \mathrm{P}_y)$ or simply $\Delta(\mathrm{P}_x, \mathrm{P}_y)$ when this causes no confusion.

A standard and very useful fact about the statistical distance is the following. For any randomized algorithm $\mathcal{A}$ that takes as input the variable x drawn from P_x, if instead we run $\mathcal{A}$ on y drawn from some distribution P_y, then the probability of success of $\mathcal{A}$ changes by at most the statistical distance between P_x and P_y:

Fact 2.4.1 ([SV03, Fact 2.4])**.** *For any randomized algorithm $\mathcal{A}$ with random coins C taking as input $x \in \mathcal{D}$, for any distributions P_x and P_y over the domain $\mathcal{D}$, we have that*

$$\left| \Pr_{\substack{x \leftarrow \mathrm{P}_x \\ C}} [\mathcal{A}(x) = 1] - \Pr_{\substack{y \leftarrow \mathrm{P}_y \\ C}} [\mathcal{A}(y) = 1] \right| \le \mathrm{TV}(\mathrm{P}_x, \mathrm{P}_y).$$

Also, we use the following triangle inequality on statistical distance. For a pdf P_x over domain $\mathcal{D}$ and $k \in \mathbb{N}$, define a new pdf $(\mathrm{P}_x)^k = \mathrm{P}_x \times \ldots \times \mathrm{P}_x$, i.e., it is the pdf of the new random variable $x' \in \mathcal{D}^k$ where each of the k coordinates is chosen from the distribution P_x i.i.d.

Fact 2.4.2 ([SV03, Fact 2.3]). *Consider variables x, y from some domain $\mathcal{D}$, and let $\mathrm{P}_x, \mathrm{P}_y$ be their pdfs. Fix $k \in \mathbb{N}$. Define new random variables $x', y' \in \mathcal{D}^k$ with pdfs $(\mathrm{P}_x)^k$ and $(\mathrm{P}_y)^k$ respectively. Then, we have*

$$\mathrm{TV}(x', y') \leq k \cdot \mathrm{TV}(x, y).$$

Concentration Bounds. We use the following standard bounds; see [MR95] for reference.

Fact 2.4.3 (Markov Bound). *Let X be a positive random variable. Then, for any $t > 0$, we have that $\Pr_X[X \geq t] \leq \mathbb{E}[X]/t$.*

Fact 2.4.4 (Chernoff Bound). *Let X_i, $i \in [n]$, be i.i.d. random Poisson trials with $\mathbb{E}[X_i] = \mu$ for some $\mu \in (0,1)$. Then, for any $\epsilon \in (0,1)$, we have $\Pr[|\sum X_i - \mu n| > \epsilon \cdot \mu n] \leq 2e^{-\mu n \epsilon^2/2}$.*

Also, for $\delta > 6$, we have $\Pr[\sum X_i \geq \delta \cdot \mu n] \leq 2^{-\delta \mu n}$.

Gaussian distribution. We will often use the *Gaussian distribution* (or *normal distribution*), parametrized by variance σ^2. The pdf of the Gaussian distribution with variance σ^2 is $N(0, \sigma^2) = \frac{1}{\sqrt{2\pi}\sigma} e^{-x^2/(2\sigma^2)}$. One standard fact about Gaussians is the following.

Fact 2.4.5. *Let x and y be random variables distributed as Gaussians with variances σ_1^2 and σ_2^2 respectively. Then the new random variable $x + y$ is distributed as a Gaussian with variance $\sigma_1^2 + \sigma_2^2$.*

A *d-dimensional Gaussian distribution* is the distribution of d-dimensional random variable, where each coordinate is drawn i.i.d. from the Gaussian distribution. The d-dimensional Gaussian distribution is spherically symmetric.

Jensen's inequality. Consider some continuous function $f : [a, b] \to \mathbb{R}$ that is concave. Then Jensen's inequality states:

Fact 2.4.6 (Jensen's inequality). *Let x be random variable on the interval $[a, b]$. Then for any concave function f, we have $\mathbb{E}_x[f(x)] \leq f(\mathbb{E}_x[x])$.*

In particular, we note that $f(x) = x \ln 1/x$ is concave on the interval $[0, 1]$.

Part I

Algorithms

Chapter 3

Locality Sensitive Hashing in the Euclidean Space

In this chapter, we present a new algorithm for solving the NN problem in the d-dimensional Euclidean space. Our algorithm achieves dn^{ρ} query time and $dn^{1+\rho}$ space and preprocessing, where $\rho(c) = 1/c^2 + o(1)$. This significantly improves over the earlier running time of [DIIM04]. In particular, for $c = 2$, our exponent tends to 0.25, while the exponent in [DIIM04] was around 0.45.

Our new algorithm falls into the class of algorithms based on Locality-Sensitive Hashing (LSH) scheme, introduced in [IM98], and is near-optimal in this class of algorithms, as proven by [MNP06, PTW08].

In the new algorithm, the convergence of the exponent to the $1/c^2$ limit is rather slow. To be more precise: the running time of the algorithm is bounded by the formula

$$t^{O(t)} n^{1/c^2 + O(\log t)/\sqrt{t}}$$

where t is a parameter chosen to minimize the expression. The $t^{O(t)}$ factor appears due to the fact that our algorithm exploits certain configurations of points in a t-dimensional space; the "quality" of the configurations increases with t. One can observe that the parameter t needs to be somewhat large for the exponent to be competitive against the earlier bounds. But then the factor $t^{O(t)}$ becomes very large, erasing the speedup gained from the improved exponent (unless n is really large).

To overcome this difficulty, we give a modification of the algorithm to make it efficient for more moderate values of n. Specifically, we replace the aforementioned configurations of points by known constructions of "nice" point-sets in specific dimensions. In particular, by utilizing the Leech Lattice [Lee67] in 24 dimensions, we obtain an algorithm with exponent $\rho(c)$ such that $\rho(2) \leq 0.37$, while the leading term in the running time is reduced to only few hundred. Moreover, if the dimension d does not exceed 24, the exponent is reduced[1] further, and we achieve $\rho(2) \leq 0.27$. The leading term in the running time remains the same.

Finally, we show that we can modify our algorithm to obtain a data structure with a near-linear space of $O(dn + n \log^{O(1)} n)$ and $dn^{O(1/c^2)}$ query time. This improves over the earlier bound of $dn^{O(1/c)}$ due to [Pan06].

[1] An astute reader will observe that if *both* the dimension d and approximation factor c are fixed constants, one can obtain a data structure with *constant* query time, essentially via table lookup. However, this approach leads to "big-Oh" constants that are exponential in the dimension, which defeats our goal of achieving a practical algorithm.

Before presenting our algorithms, we give an overview of the Locality-Sensitive Hashing scheme and how it is used to obtain NN algorithms. The results from this chapter have previously appeared in [AI06b, AI08b, ADI+06].

3.1 LSH: A Primer

The LSH scheme relies on the existence of *locality-sensitive hash functions.* Let $\mathcal{H}$ be a family of hash functions mapping $\mathbb{R}^d$ to some discrete universe U. For some arbitrary two points p and q, consider the process in which we choose a function h from $\mathcal{H}$ uniformly at random, and analyze the probability that $h(p) = h(q)$. The family $\mathcal{H}$ is called *locality sensitive* (with proper parameters) if it satisfies the following condition.

Definition 3.1.1 (Locality-sensitive hashing)**.** *A family $\mathcal{H}$ is called* (R, cR, P_1, P_2)-sensitive *if for any two points $p, q \in \mathbb{R}^d$*

- *if $\|p - q\| \leq R$ then $\Pr_{\mathcal{H}}[h(q) = h(p)] \geq P_1$,*
- *if $\|p - q\| \geq cR$ then $\Pr_{\mathcal{H}}[h(q) = h(p)] \leq P_2$.*

In order for a locality-sensitive hash (LSH) family to be useful, it has to satisfy $P_1 > P_2$.

To illustrate the concept, consider the following example. Assume that the data points are *binary*, that is, each coordinate is either 0 or 1. In addition, assume that the distance between points p and q is computed according to the Hamming distance. In this case we can use a particularly simple family of functions $\mathcal{H}$ which contains all projections of the input point on one of the coordinates. That is, $\mathcal{H}$ contains all functions h_i from $\{0,1\}^d$ to $\{0,1\}$ such that $h_i(p) = p_i$, for $i = 1 \dots d$. Choosing one hash function h uniformly at random from $\mathcal{H}$ means that $h(p)$ returns a random coordinate of p (note however, that different applications of h return the *same* coordinate of the argument).

To see that the family $\mathcal{H}$ is locality-sensitive with non-trivial parameters, observe that the probability $\Pr_{\mathcal{H}}[h(p) = h(q)]$ is equal to the fraction of coordinates on which p and q agree. Therefore, $P_1 = 1 - R/d$, while $P_2 = 1 - cR/d$. As long as the approximation factor c is greater than 1, we have $P_1 > P_2$.

3.1.1 The algorithm

An LSH family $\mathcal{H}$ can be used to design an efficient algorithm for approximate near neighbor search. However, one typically cannot use $\mathcal{H}$ as is, since the gap between the probabilities P_1 and P_2 could be quite small. Instead, an "amplification" process is needed in order to achieve the desired probabilities of collision. We describe this step next, and present the complete algorithm in the Figure 3-1.

Given a family $\mathcal{H}$ of hash functions with parameters (R, cR, P_1, P_2) as in the above definition, we amplify the gap between the "high" probability P_1 and "low" probability P_2 by concatenating several functions. In particular, for parameters k and L (specified later), we choose L functions $g_j(q) = (h_{1,j}(q), \dots, h_{k,j}(q))$, where $h_{t,j} (1 \leq t \leq k, 1 \leq j \leq L)$ are chosen independently and uniformly at random from $\mathcal{H}$. These are the actual functions that we use to hash the data points.

The data structure is constructed by placing each point p from the input set into a bucket $g_j(p)$, for $j = 1, \dots, L$. Since the total number of buckets may be large, we retain only the

Preprocessing:

1. Choose L functions g_j, $j = 1, \dots L$, by setting $g_j = (h_{1,j}, h_{2,j}, \dots h_{k,j})$, where $h_{1,j}, \dots h_{k,j}$ are chosen at random from the LSH family $\mathcal{H}$.
2. Construct L hash tables, where, for each $j = 1, \dots L$, the j^{th} hash table contains the dataset points hashed using the function g_j.

Query algorithm for a query point q:

1. For each $j = 1, 2, \dots L$

   i) Retrieve the points from the bucket $g_j(q)$ in the j^{th} hash table.

   ii) For each of the retrieved point, compute the distance from q to it, and report the point if it is a correct answer (cR-near neighbor for Strategy 1, and R-near neighbor for Strategy 2).

   iii) (*optional*) Stop as soon as the number of reported points is more than L'.

Figure 3-1: Preprocessing and query algorithms of the basic LSH algorithm.

non-empty buckets by resorting to (standard) hashing[2] of the values $g_j(p)$. In this way, the data structure uses only $O(nL)$ memory cells; note that it suffices that the buckets store the pointers to data points, not the points themselves.

To process a query q, we scan through the buckets $g_1(q), \dots, g_L(q)$, and retrieve the points stored in them. After retrieving the points, we compute their distances to the query point, and report any point that is a valid answer to the query. Two concrete scanning strategies are possible:

1. Interrupt search after finding the first L' points (including duplicates), for some parameter L'.

2. Continue search until all points from all buckets are retrieved; no additional parameter is required.

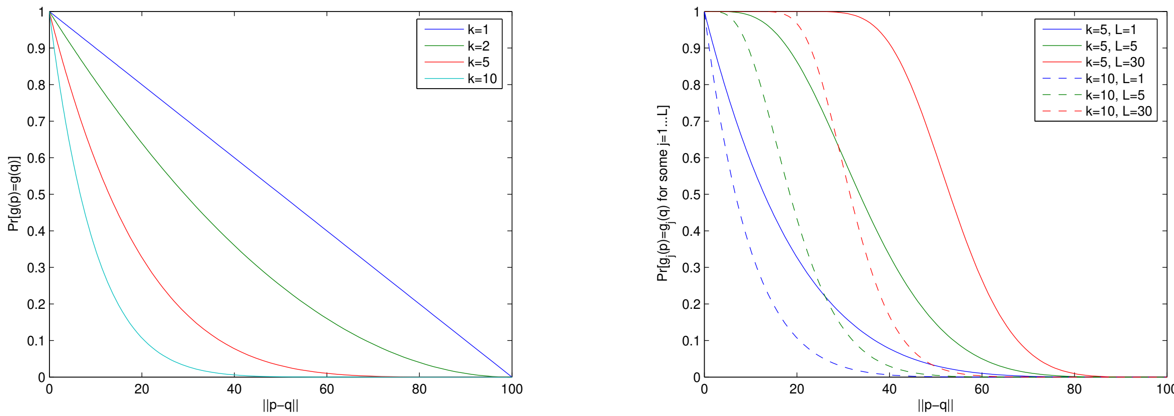
The two strategies lead to different behaviors of the algorithms. In particular, Strategy 1 solves the (c, R)-near neighbor problem, while Strategy 2 solves the R-near neighbor reporting problem.

Strategy 1: It is shown in [IM98, GIM99] that the first strategy, with $L' = 3L$, yields a solution to the *randomized c-approximate R-near neighbor problem*, with parameters R and δ, for some constant failure probability $\delta < 1$. To obtain that guarantee, it suffices to set L to $\Theta(n^\rho)$, where $\rho = \frac{\ln 1/P_1}{\ln 1/P_2}$ [GIM99], and k to $\log_{1/P_2} n$. Note that this implies that the algorithm runs in time proportional to n^ρ, which is sublinear in n if $P_1 > P_2$. For example, if we use the hash functions for binary vectors mentioned earlier, we obtain $\rho = 1/c$ [IM98, GIM99].

Strategy 2: The second strategy enables us to solve the *randomized R-near neighbor reporting problem* (as defined in Section 1.2). The value of the failure probability δ depends on the choice of the parameters k and L. Conversely, for each δ one can provide parameters k and L so that the error probability is smaller than δ. The query time is also dependent on k and L: it could be as high as $\Theta(n)$ in the worst case, but for many natural data sets a proper choice of parameters results in a sublinear query time.

The details of the analysis are as follows. Let p be any R-neighbor of q, and consider any parameter k. For any function g_i, the probability that $g_i(p) = g_i(q)$ is at least P_1^k.

[2]See [CLRS01] for more details on hashing.



(a) The probability that $g_j(p) = g_j(q)$ for a fixed j. Graphs are shown for several values of k. In particular, the blue function ($k = 1$) is the probability of collision of points p and q under a single random hash function h from the LSH family.

(b) The probability that $g_j(p) = g_j(q)$ for some $j = 1 \dots L$. The probabilities are shown for two values of k and several values of L. Note that the slopes are sharper when k is higher.

Figure 3-2: The graphs of the probability of collision of points p and q as a function of the distance between p and q for different values of k and L. The points p and q are $d = 100$ dimensional binary vectors under the Hamming distance. The LSH family $\mathcal{H}$ is the one described in Section 3.1.

Therefore, the probability that $g_i(p) = g_i(q)$ for *some* $i = 1 \dots L$ is at least $1 - (1 - P_1^k)^L$. If we set $L = \log_{1-P_1^k} \delta$ so that $(1 - P_1^k)^L \le \delta$, then any R-neighbor of q is returned by the algorithm with probability at least $1 - \delta$.

How should the parameter k be chosen? Intuitively, larger values of k lead to a larger gap between the probabilities of collision for "close" points and "far" points; the probabilities are P_1^k and P_2^k, respectively (see Figure 3-2 for an illustration). The benefit of this amplification is that the hash functions are more "selective". At the same time, if k is "large" then P_1^k is "small", which means that L must be sufficiently "large" to ensure that an R-near neighbor collides with the query point at least once.

To obtain worst-case guarantees, we choose k and L to be $k = \log_{1/P_2} n$ and $L = n^\rho$, where $\rho = \frac{\log 1/P_1}{\log 1/P_2}$. The query time becomes $O(n^\rho k \cdot \tau)$, where τ is the time to compute $h \in \mathcal{H}$, and the preprocessing time becomes $O\left(n^{1+\rho} k \cdot \tau\right)$.

In practice, we may want to choose parameter k (and thus L) differently. One such approach to choosing k was introduced in the E^{2}LSH package [AI05]. There, the data structure optimized the parameter k as a function of the data set and a set of sample queries. Specifically, given the data set, a query point, and a fixed k, one can estimate precisely the expected number of collisions and thus the time for distance computations, as well as the time to hash the query into all L hash tables. The sum of the estimates of these two terms is the estimate of the total query time for this particular query. E^{2}LSH chooses k that minimizes this sum over a small set of sample queries.

3.2 Near-Optimal LSH for Euclidean Space

In this section we present a new LSH family, yielding an algorithm with query time exponent[3]

$$\rho(c) = 1/c^2 + O\left(\frac{\log\log n}{\log^{1/5} n}\right).$$

For large enough n, the value of $\rho(c)$ tends to $1/c^2$. We then obtain the following algorithm.

Theorem 3.2.1. *There exists an algorithm solving c-NN problem in d-dimensional Euclidean space that achieves $dn^{1/c^2+o(1)}$ query time and $dn^{1+1/c^2+o(1)}$ space and preprocessing.*

We first give a high-level intuition of our new LSH family. We obtain our result by carefully designing a family of locality-sensitive hash functions in ℓ_2. The starting point of our construction is the "line partitioning" method of [DIIM04]. There, a point p was mapped into $\mathbb{R}^1$ using a random projection. Then, the line $\mathbb{R}^1$ was partitioned into intervals of length w, where w is a parameter. The hash function for p returned the index of the interval containing the projection of p.

An analysis in [DIIM04] showed that the query time exponent has an interesting dependence on the parameter w. If w tends to infinity, the exponent tends to $1/c$, which yields no improvement over [IM98, GIM99]. However, for small values of w, the exponent lies slightly below $1/c$. In fact, the unique minimum exists for each c.

In our new algorithm we utilize a "multi-dimensional version" of the aforementioned approach. Specifically, we first perform random projection into $\mathbb{R}^t$, where t is super-constant, but relatively small (i.e., $t = o(\log n)$). Then we partition the space $\mathbb{R}^t$ into cells. The hash function function returns the index of the cell which contains projected point p.

The partitioning of the space $\mathbb{R}^t$ is somewhat more involved than its one-dimensional counterpart. First, observe that the natural idea of partitioning using a grid does not work. This is because this process roughly corresponds to hashing using concatenation of several one-dimensional functions (as in [DIIM04]). Since the LSH algorithms perform such concatenation anyway, grid partitioning does not result in any improvement. Instead, we use an approach similar to the method of "ball partitioning", introduced in [CCG+98] in the context of embeddings into tree metrics. The partitioning is obtained as follows. We create a sequence of balls $B_1, B_2 \ldots$, each of radius w, with centers chosen independently "at random". Each ball B_i then defines a cell, containing points $B_i \setminus \cup_{j<i} B_j$.

In order to apply this method in our context, we need to take care of a few issues. First, locating a cell containing a given point could require enumeration of all balls, which would take an unbounded amount of time. Instead, we show that one can simulate the above procedure by replacing each ball by a "grid of balls". It is not difficult then to observe that a finite (albeit exponential in t) number U of such grids suffices to cover all points in $\mathbb{R}^t$. An example of such partitioning (for $t = 2$ and $U = 5$) is given in Figure 3-3.

The second and the main issue is the choice of w. Again, it turns out that for large w, the method yields only the exponent of $1/c$. Specifically, it was shown in [CCG+98] that for any two points $p, q \in \mathbb{R}^t$, the probability that the partitioning separates p and q is at most $O\left(\sqrt{t} \cdot \|p-q\|/w\right)$. This formula can be showed to be tight for the range of w where it makes sense as a lower bound, that is, for $w = \Omega\left(\sqrt{t} \cdot \|p-q\|\right)$. However, as long as the separation probability depends linearly on the distance between p and q, the exponent ρ is

[3]A tighter analysis gives $\rho(c) = 1/c^2 + O(\log\log n/\log^{1/3} n)$ [AI06a]. Here we present a simpler bound for a more simplified exposition.

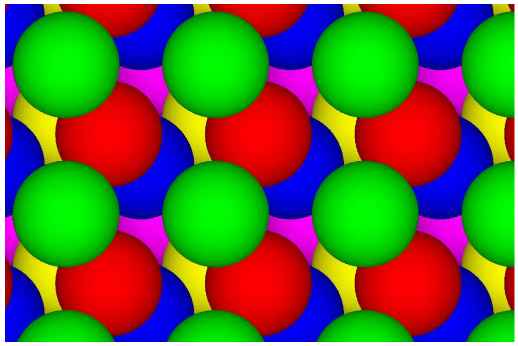


Figure 3-3: An illustration of the ball partitioning of the 2-dimensional space.

still equal to $1/c$. Fortunately, a more careful analysis shows that, as in the one-dimensional case, the minimum is achieved for finite w. For that value of w, the exponent tends to $1/c^2$ as t tends to infinity.

We observe that we can assume that $R = 1$ for the rest of this chapter (by rescaling the coordinates).

3.2.1 Algorithm description

We first describe an "ideal" LSH family for ℓ_2. This approach leads to the desired bound on ρ but the resulting hash functions have a prohibitively high time complexity. We then show how to modify this "ideal" LSH family to obtain one with a better time complexity. The final description of the LSH family is presented in the figure 3-4.

Ideal LSH family. Construct a hash function $\tilde{h}$ as follows. Consider G^d, a regular infinite grid of balls in $\mathbb{R}^d$: each ball has radius w and has the center at $4w \cdot \mathbb{Z}^d$. Let G^d_u, for positive integer u, be the grid G^d shifted uniformly at random; in other words, $G^d_u = G^d + s_u$, where $s_u \in [0, 4w]^d$. Now we choose as many G^d_u's as are needed to cover the entire space $\mathbb{R}^d$ (i.e., until each point from $\mathbb{R}^d$ belongs to at least one of the balls). Suppose we need U such grids to cover the entire space with high probability.

We define $\tilde{h}$ on a point p as a tuple $(u, x_1, x_2, ...x_d)$, $u \in [1, U]$ and $(x_1, ...x_d) \in G^d_u$. The tuple $(u, x_1, x_2, ...x_d)$ specifies the ball which contains the point p: $p \in B((x_1, x_2, \ldots x_n), w)$. If there are several balls that contain p, then we take the one with the smallest value u. Computing $\tilde{h}(p)$ can be done in $\tau = O(U)$ time: we iterate through all $G^d_1, G^d_2, ...G^d_U$, and find the first G^d_u such that p is inside a ball with the center from G^d_u.

Intuitively, this family satisfies our locality-sensitive definition: the closer are the points p, q, the higher is the probability that p, q belong to the same ball. Indeed, if we choose a suitable radius $w \geq 1/2$, then we will get $L = n^\rho = n^{1/c^2+o(1)}$.

However, the deficiency of this family is that the time to compute $\tilde{h}(p)$ might be too large if $d = \Omega(\log n)$ since we need to set $U = \Omega(2^d)$ (see Lemma 3.2.2). We show how to circumvent this deficiency next.

Actual LSH family. Our actual construction utilizes the "ideal" family described above, while introducing an additional step, necessary to reduce U, the number of grids covering the space. The algorithm is given in Figure 3-4.

Initialization of a hash function $h \in \mathcal{H}$

1. For $u = 1$ to U, choose a random shift $s_u \in [0, 4w]^t$, which specifies the grid $G_u^t = G^t + s_u$ in the t-dimensional Euclidean space.
2. Choose a matrix $A \in M_{t,d}$, where each entry A_{ij} is distributed according to the normal distribution $N(0, 1)$ times a scaling factor, $\frac{1}{\sqrt{t}}$. The matrix A represents a random projection from $\mathbb{R}^d$ to $\mathbb{R}^t$.

Computing $h()$ **on a point** $p \in \mathbb{R}^d$

1. Let $p' = Ap$ be the projection of the point p onto the t-dimensional subspace given by A.
2. For each $u = 1, 2, \ldots U$
    3. Check whether $B(p', w) \cap G_u^t \neq \emptyset$, i.e., whether there exist some $(x_1, x_2, \ldots x_t) \in G_u^t$ such that $p \in B((x_1, x_2, \ldots x_t), w)$.
    4. Once we find such $(x_1, x_2, \ldots x_t)$, set $h(p) = (u, x_1, x_2, \ldots x_t)$, and stop.
5. Return 0^{t+1} if we do not find any such ball.

Figure 3-4: Algorithms for initializing a hash function h from the LSH hash family, and for computing $h(p)$ for a point $p \in \mathbb{R}^d$.

To reduce U, we project $\mathbb{R}^d$ to a lower-dimensional space $\mathbb{R}^t$ via a random dimensionality reduction. The parameter t is $o(\log n)$, such that factors exponential in t are $o(n)$. After performing the projection, we choose the grids $G_1^t, G_2^t, \ldots G_U^t$ in the lower-dimensional space $\mathbb{R}^t$. Now, to compute $h(p)$, we compute the projection of p onto the lower dimensional space $\mathbb{R}^t$, and process the projected point as described earlier. In short, the actual hash function is $h(p) = \tilde{h}(Ap)$, where A is a random matrix representing the dimensionality reduction mapping, and $\tilde{h}$ works in the t-dimensional space. Note that τ becomes $\tau = O(dt) + O(U_t)$ corresponding to the projection and the bucket-computation stages respectively.

3.2.2 Analysis of the LSH Family

We now prove the main theorem of this chapter, Theorem 3.2.1. The theorem relies on two lemmas.

The first lemma bounds the number of grids G^d needed to cover the entire space $\mathbb{R}^d$, for any dimension d. This number impacts directly the query time of our algorithm.

Lemma 3.2.2. *Consider a d-dimensional space $\mathbb{R}^d$, and fix some $\delta > 0$. Let G^d be a regular infinite grid of balls of radius w placed at coordinates $\sigma w \cdot \mathbb{Z}^d$, where $2 \leq \sigma \leq d$. Define G_u^d, for $u \in \mathbb{N}$, as $G_u^d = G^d + s_u$, where $s_u \in [0, \sigma w]^d$ is a random shift of the grid G^d. If $U_d = 2^{O(d \log d)} \log 1/\delta$, then the grids $G_1^d, G_2^d, \ldots G_{U_d}^d$ cover the entire space $\mathbb{R}^d$, with probability ar least $1 - \delta$.*

The second lemma is the key technical ingredient to our result, and it bounds the parameter ρ.

Lemma 3.2.3. *Consider the hash function h described in the Figure 3-4, and let p, q be some points in $\mathbb{R}^d$. Let P_1 be the probability that $h(p) = h(q)$ given that $||p-q|| \leq 1$, and let P_2 be the probability that $h(p) = h(q)$ given that $||p - q|| \geq c$. Then, for $w = t^{1/3}$, we obtain $\rho = \frac{\log 1/P_1}{\log 1/P_2} = 1/c^2 + O\left(1/t^{1/4}\right)$. Moreover, $P_2 \geq \exp\left[-O(t^{1/3})\right]$.*

We remark that, one can also obtain a tighter bound on the second-order term of ρ, namely $\rho = 1/c^2 + O\left(\frac{\log t}{\sqrt{t}}\right)$ (see [AI06a]). We omit that derivation from this thesis as it does not lead to a better exponent in Theorem 3.2.1, and improves only the second order terms.

Before proving the two lemma, we now show how to complete the proof of Theorem 3.2.1.

Proof of Theorem 3.2.1. The result follows by using the LSH family in Figure 3-4 with the general LSH scheme described in Section 3.1. By Lemma 3.2.3, for $t = \log^{4/5} n$, we have $\rho = 1/c^2 + O\left(\frac{1}{\log^{1/5} n}\right)$. Furthermore, k can be bounded as

$$k = \frac{\log n}{\log 1/P_2} \leq \frac{\log n}{\Omega(t^{1/3})} \leq O(\log n).$$

Finally, by Lemma 3.2.2 for $\sigma = 4$, we have that $\tau = O(dt) + O(U^t) = O(dt) + O\left(2^{t \log t} \log n\right) = O(dt) + 2^{O(\log^{4/5} n \log\log n)} \log n = O(dn^{o(1)})$. The theorem follows. □

We now prove Lemmas 3.2.2 and 3.2.3.

Proof of Lemma 3.2.2. This lemma is a relatively standard random packing argument (see, e.g., [PA95]). For completeness, we give a version of the argument below.

The intuition behind the lemma is simple. First, observe that the entire space is covered if and only if the hypercube $[0, \sigma w]^d$ is covered by grids G_u^d (due to the regularity of the grids). Now, the intersection of the hypercube $[0, \sigma w]^d$ and a grid G_u^d has volume precisely $\mathrm{Vol}^d(w) = S_d \cdot w^d/d$, and covers a fraction of $2^{-O(d \log d)}$ of the hypercube. If the grids did not intersect, we would need only $2^{O(d \log d)}$ grids to cover the space. In general, a grid will cover an expected of $2^{-O(d \log d)}$ fraction of the hypercube, unless most of the hypercube has been already covered (in which case, we are close to being done anyways).

We argue formally as follows. We partition the hypercube $[0, \sigma w]^d$ into smaller "micro-cubes" and prove that each of them is covered with a high enough probability. Specifically, we partition the hypercube $[0, \sigma w]^d$ into smaller micro-cubes, each of size $\frac{w}{\sqrt{d}} \times \frac{w}{\sqrt{d}} \cdots \times \frac{w}{\sqrt{d}}$. There are $N = \frac{(\sigma w)^d}{(w/\sqrt{d})^d} = (\sigma\sqrt{d})^d$ such micro-cubes in total. Let x be the probability that a micro-cube is covered by one grid G_u^d. Then $x \geq \frac{(w/\sqrt{d})^d}{(\sigma w)^d} = 1/N$ because, for a micro-cube to be covered, it suffices that the center of the ball $B(0^d + s_u, w)$ falls inside the micro-cube, which happens with probability $1/N$. Furthermore, if x_U is the probability that a micro-cube is covered by any of the U_d grids G_u^d, then $x_U \geq 1 - (1-x)^{U_d}$.

Thus, we can compute the probability that there exists at least one uncovered micro-cube, which is also the probability that the entire $[0, \sigma w]^d$ hypercube is uncovered. Set $U_d = aN(\log n + \log N)$ for a suitable constant a. Using union bound, we obtain that the probability that the entire hypercube is not covered is at most

$$N\,(1-x)^{U_d} \leq N(1-1/N)^{U_d} \leq N(1-1/N)^{aN(\log n + \log N)} \leq N 2^{-\log n - \log N} \leq 1/n.$$

Concluding, with probability at least $1 - 1/n$ we cover the entire space with the grids $G_1^d, \ldots G_{U_d}^d$, if we choose $U_d = O(N(\log n + \log N)) = 2^{O(d \log d)} \log n$. □

We now proceed to proving Lemma 3.2.3.

Proof. Fix some points $p, q \in \mathbb{R}^d$ at distance $\Delta = \|p - q\|$. The proof proceeds in three stages. First we show that the effects of the dimensionality reduction into $\mathbb{R}^t$ are negligible. Second, we relate the probability that $h(p) = h(q)$ to the volume of a certain cap of a sphere in $\mathbb{R}^t$. Finally, using standard approximations of high-dimensional spheres, we compute the probabilities P_1, P_2 and then the value ρ.

The points p and q are projected into points $p' = Ap$ and $q' = Aq$, where $p', q' \in \mathbb{R}^t$. For $\Delta' = \|p' - q'\|$, the distortion of the projection is $\frac{\Delta'}{\Delta} = \frac{\|p'-q'\|}{\|p-q\|}$ and we consider it to be "high" when it is either greater than $1+\epsilon$ or smaller than $1-\epsilon$, for some ϵ. The probability of high distortion is upper bounded by $f = \exp\left[-\Omega(\epsilon^2 t)\right]$, using standard bound on distortion under random projections, namely Fact 2.3.2. We use $\epsilon = 1/\sqrt[4]{t}$, in which case, we obtain the following bound.

Claim 3.2.4. *When $\epsilon = t^{-1/4}$, we have $\Delta'/\Delta \in [1-\epsilon, 1+\epsilon]$ with probability at least $1 - f = 1 - \exp\left[-\Omega(\sqrt{t})\right]$.*

Now consider two points $p', q' \in \mathbb{R}^t$ at distance Δ'. The sequence of grids $G_1^t, G_2^t, \ldots, G_U^t$ covers the entire space $\mathbb{R}^t$ with very high probability, and thus p' and q' are inside some balls. Let G_u^t be the first grid such that either p' or q' is inside some ball $B(x, w)$ which belongs to the grid G_u^t. The position of this ball completely determines whether $h(p) = h(q)$ or not. In particular, $h(p) = h(q)$ when both p' and q' are inside $B(x, w)$, or, equivalently, $x \in B(p', w) \cap B(q', w)$. Otherwise, $h(p) \neq h(q)$ when exactly one of p', q' is inside $B(x, w)$, or equivalently, x is inside the symmetric difference of $B(p', w)$ and $B(q', w)$). Thus, the probability that $h(p) = h(q)$ is exactly the ratio of the volume of the intersection of two balls, $B(p', w) \cap B(q', w)$, over the volume of the union of two balls, $B(p', w) \cup B(q', w)$. See the Figure 3-5 for an illustration. We deduce the following expression for probability that $h(p) = h(q)$. As defined in the preliminaries, $C(\Delta'/2, w)$ denotes the cap volume, and $I(\Delta'/2, w)$ denotes the cap volume divided by the ball volume.

$$\Pr_h[h(p) = h(q)] = \frac{|B(p', w) \cap B(q', w)|}{|B(p', w) \cup B(q', w)|} = \frac{2C(\Delta'/2, w)}{2\,\mathrm{Vol}^t(w) - 2C(\Delta'/2, w)} = \frac{I(\Delta'/2, w)}{1 - I(\Delta'/2, w)} \tag{3.1}$$

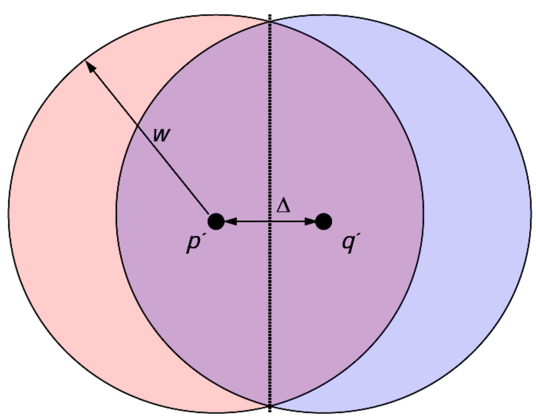


Figure 3-5: A diagram of the balls $B(p', w)$ and $B(q', w)$. The red zone is the locus of the centers of grid balls capturing p' but not q'; blue zone — for those capturing q' but not p'; magenta (intersection) zone — for those capturing both p' and q'. The magenta zone is composed of exactly two caps (separated by the dotted line), whose volume we denote by $C(\Delta'/2, w)$.

The main remaining step is to evaluate the quantity $I(\Delta'/2, w)$. We give below a somewhat informal calculations below, which we formalize later on. The quantity $I(\Delta'/2, w)$ can also be viewed as the following probability: "draw a random point x from the ball of radius w, what is the probability that the first coordinate of x is greater than $\Delta'/2$?" A standard fact in high-dimension geometry is that the distribution of a random point x from a ball of radius w in $\mathbb{R}^t$ is well approximated by the t-dimensional normal distribution, when t is sufficiently high. A t-dimensional normal distribution is a product of t independent normal distributions, in particular implying that the first coordinate of x is distributed as a normal distribution. Thus, essentially, $I(\Delta'/2, w)$ may be approximated by the tail of a normal distribution, in particular a tail starting at distance $\Delta'/2w$ from the origin. Now, standard facts about tails of normal distribution tell us that this tail is roughly equal to $e^{-A\cdot\Delta'^2}$, where A is some constant (dependent on w and the dimension t). Finally, ρ is the ratio of the exponents of P_1 and P_2, the collision probabilities when $\Delta' = 1$ and $\Delta' = c$ respectively. Hence, we obtain a value of ρ which is roughly equal to $1/c^2$.

We now formalize the calculations from above. We use the estimate of $I(\Delta'/2, w)$ from Fact 2.3.1. We compute P_1 and P_2 separately. We use the fact that $\Delta'/\Delta \in [1-\epsilon, 1+\epsilon]$ with probability at least $1 - f = 1 - \exp\left[-\Omega(\sqrt{t})\right]$ by Claim 3.2.4. We compute P_1 from Eqn. (3.1) by setting $\Delta = 1$, in which case also $\Delta' \le 1 + \epsilon$ with probability at least $1 - f$:

$$P_1 \ge I(\Delta'/2, w) - f \ge \frac{A_l}{\sqrt{t}} \exp\left[-\frac{t}{2} \cdot \frac{(\Delta'/2w)^2}{1-(\Delta'/2w)^2}\right] - f \ge \frac{A_l}{2\sqrt{t}} \exp\left[-t^{1/3} \cdot \frac{(1+\epsilon)^2(1+1/w^2)}{8}\right]. \tag{3.2}$$

Similarly, we compute P_2 using Fact 2.3.1 and setting $\Delta = c$, in which case $\Delta' \ge c(1-\epsilon)$ with probability at least $1 - f$:

$$P_2 \le 2I(\Delta'/2, w) + f \le 2\exp\left[-\frac{t}{2} \cdot (\Delta'/2w)^2\right] + f \le 3\exp\left[-t^{1/3} \cdot c^2 \cdot \frac{(1-\epsilon)^2}{8}\right]. \tag{3.3}$$

Finally, we get the following bound on $\rho(c) = \frac{\log 1/P_1}{\log 1/P_2}$:

$$\rho(c) \le \frac{t^{1/3} \cdot \frac{(1+\epsilon)^2(1+1/w^2)}{8} + O(\log t)}{t^{1/3} \cdot c^2 \cdot \frac{(1-\epsilon)^2}{8} - O(1)} \le \frac{1}{c^2} + O(\epsilon) + O(1/w^2) + O\left(\frac{\log t}{t^{1/3}}\right) \le \frac{1}{c^2} + O(1/t^{1/4}). \tag{3.4}$$

This completes the proof of Lemma 3.2.3. □

3.3 Lattice-based LSH Family

In this section we describe a practical variant of the above LSH family based on lattices in Euclidean spaces. Although, theoretically, these families are not asymptotically better than the ones described earlier, they are likely to perform better in practice, due to the much lower "big-Oh" constants.

Our goal is to introduce a different (but related) partitioning method that avoids the $t^{O(t)}$ factor. Specifically, we use tessellations induced by (randomly shifted) Voronoi diagrams of *fixed* t-dimensional point constellations which have the following two nice properties:

- The closest constellation point to a given point can be found efficiently, and
- The exponent ρ induced by the constellation is as close to $1/c^2$ as possible.

The partitioning is then implemented by randomly projecting the points into $\mathbb{R}^t$, and using the Voronoi diagram. We discovered that a constellation in 24 dimensions known as the Leech Lattice [Lee67] satisfies the above properties quite well. First, the nearest point in the lattice can be found by using a (bounded) decoder of [AB96] which perform only 519 floating point operations per decoded point. Second, the exponent $\rho(c)$ guaranteed by that decoder is quite attractive: for $c = 2$ the exponent $\rho(2)$ is less than 0.37. The intuitive reason for that is that the Leech Lattice is a "very symmetric" constellation, and thus its Voronoi cells are very "round". Moreover, if the dimension d does not exceed 24, then we can skip the dimensionality reduction part. In that case we obtain $\rho(2) \leq 0.27$, while the leading term in the running time remains the same.

We start by presenting the general lattice-based approach. Then, we give an algorithm based on a concrete 24-dimensional lattice, called the Leech Lattice [Lee67]. For the Leech lattice-based algorithm, we include the actual values of the resulting exponent ρ, the main indicator of the performance.

3.3.1 Lattices in arbitrary dimension

The algorithm in this section uses an arbitrary lattice in some t-dimensional space. An example of a t-dimensional lattice is the regular grid of points in $\mathbb{R}^t$, although, as mentioned in Section 3.2, it does not serve well our purposes. For a given lattice, we need an efficient lattice decoding function, to which we refer as $\textsc{LatticeDecode}(x)$. The function $\textsc{LatticeDecode}(x)$ takes as input a point $x \in \mathbb{R}^t$ and returns the lattice point that is the closest to x.

Given a specific lattice with a decoding function $\textsc{LatticeDecode}(x)$, an LSH function is constructed as follows (formally presented in Figure 3-6). First, if $d > t$, we choose a random projection from the d-dimensional space to the t-dimensional space, which we represent as a matrix A of dimension $t \times d$. If $d \leq t$, then, instead, we choose a random *rotation* in the t-dimensional space, which we also represent as a matrix A of dimension $t \times d$ (here, A is equal to the first d columns of an random orthonormal matrix of dimension $t \times t$). Finally, we choose a random *translation* in the t-dimensional space, which we represent as a vector T of dimension $t \times 1$. The values of A and T identify an LSH function.

For an LSH function h specified by A and T, we define $h(p)$ as being $h(p) = \textsc{LatticeDecode}(A \cdot p + T)$. Or, in words, for $p \in \mathbb{R}^d$, we first project p into $\mathbb{R}^t$ using A (or rotate it if $d \leq t$); then, we translate the projection using T; and, finally, we find the closest point in lattice using $\textsc{LatticeDecode}$. The output of $\textsc{LatticeDecode}$ gives the value of $h(p)$.

The performance of the resulting LSH scheme depends heavily on the choice of the lattice. Intuitively, we would like a lattice that lives in $\mathbb{R}^t$ for high t, is "dense"[4], and has a fast decoding function $\textsc{LatticeDecode}$. With a higher t, the dimensionality reduction is more accurate. A "denser" lattice gives a sharper difference in collision probabilities of close and far points.

3.3.2 Leech Lattice

In this section, we focus on a particular lattice in 24 dimensional space, the Leech Lattice [Lee67]. We give numerical values for the ρ when we use the Leech Lattice in the algorithm from Figure 3-6 with a specific decoder described below.

[4] A measure of "density" is, for example, the density of hypersphere packing induced by the lattice. The density of hypersphere packing is the percent of the space that is covered by non-overlapping balls centered at lattice points.

Initialization of a hash function $h \in \mathcal{H}$

1. If $d > t$, choose a random projection from d-dimensional space to t-dimensional space. The projection is represented by a $t \times d$ matrix A, where each element A_{ij} is distributed according to the normal distribution $N(0, 1)$ times a scaling factor of $\frac{1}{\sqrt{t}}$.
2. If $d \leq t$, choose a random rotation in the t-dimensional space. The rotation is represented by the matrix A, which is equal to the first d coordinates of an $t \times t$ orthonormal matrix.
3. Choose a random translation in the t-dimensional space. The translation is represented by a vector $T \in M_{t,1}$.

Computing $h()$ on a point $p \in \mathbb{R}^d$

1. Let $x = A \cdot p + T$.
2. Return $\textsc{LatticeDecode}(x)$.

Figure 3-6: Algorithms for initializing an LSH function h and for computing $h(p)$ for a point $p \in \mathbb{R}^d$.

The Leech Lattice has been studied extensively (see, e.g., [CS93, CS86, ABV$^+$94, AB96]) and is known to be the lattice that gives the densest (lattice) hypersphere packing in 24 dimensions. Below, we denote the Leech Lattice by λ_{24} and call the corresponding decoding function $\textsc{LatticeDecode}_{\lambda_{24}}(x)$.

Several efficient decoders for the Leech Lattice are known (see, e.g., [ABV$^+$94, AB96, Var95]); the best of them require a few thousand floating point operations to decode one point. However, even faster decoders are known for the *bounded-distance* decoding problem. A bounded-distance decoder guarantees to return the correct result only when the query point x is sufficiently close to one of the lattice points; otherwise the decoder gives no guarantees. Note that a bounded-distance decoder yields an LSH function (albeit not necessarily as good as the perfect decoder), as long as the decoder is deterministic.

We have investigated the bounded-distance decoder of [AB96]. Their implementation uses at most 519 real operations per decoded point. Since their implementation decodes only a finite subset of the Leech Lattice, we have slightly modified the implementation to approximate the decoding of the entire Leech Lattice. We call this bounded-distance decoder $\textsc{LatticeDecode}^B_{\lambda_{24}}(x)$.

For that decoder, we computed the values of the resulting collision probabilities (for the case $d > 24$). The results are depicted in Table 3.1. The probabilities are computed using Monte-Carlo simulation with 10^7 trials. Specifically, in a trial, we generate a random point p and some other point q, such that $p-q$ is drawn from a 24-dimensional Gaussian distribution, scaled by $\frac{1}{\sqrt{24}}$ times the radius. The points p and q collide iff $\textsc{LatticeDecode}^B_{\lambda_{24}}(p) = \textsc{LatticeDecode}^B_{\lambda_{24}}(q)$. Table 3.1 summarizes the estimated probabilities of collision for different values of radii (the confidence intervals are computed with 95% accuracy). These probabilities yield values for ρ that are summarized in Table 3.2. The table shows maximum likelihood ρ and conservative ρ. The max likelihood ρ is the ratio of corresponding max likelihood values of P_1 and P_2 (from the middle column). The conservative ρ is the ratio of lowest estimate of P_1 from the confidence interval to the highest estimate of P_2 in the confidence interval.

From the table, one can observe that the Leech Lattice offers a substantial improvement over method from [DIIM04]. For example, for the approximation $c = 2$, we obtain an exponent $\rho = 0.36$, whereas [DIIM04] obtains an exponent of 0.45. However, our exponent

Radius	Est. collision probability	Confidence interval
0.7	0.0853465	[0.0853409, 0.0853521]
0.8	0.0525858	[0.0525813, 0.0525903]
0.9	0.0311720	[0.0311685, 0.0311755]
1.0	0.0177896	[0.0177869, 0.0177923]
1.1	0.0097459	[0.0097439, 0.0097479]
1.2	0.0051508	[0.0051493, 0.0051523]
1.3	0.0026622	[0.0026611, 0.0026633]
1.4	0.0013332	[0.0013324, 0.0013340]
1.5	0.0006675	[0.0006670, 0.0006681]
1.6	0.0003269	[0.0003265, 0.0003273]
1.7	0.0001550	[0.0001547, 0.0001553]
1.8	0.0000771	[0.0000769, 0.0000773]
1.9	0.0000368	[0.0000366, 0.0000370]
2.0	0.0000156	[0.0000155, 0.0000157]

Table 3.1: Probabilities of collision of two points, for $d > 24$, under the hash function described in Figure 3-6 with bounded-distance Leech Lattice decoder. The values were obtained through Monte-Carlo simulation for 10^7 trials. Confidence interval corresponds to 95% accuracy.

c	Max likelihood of ρ	Conservative ρ	Radius R
1.5	0.5563	0.5565	1.2
2.0	0.3641	0.3643	1.0

Table 3.2: The values of $\rho = \frac{\log P_1}{\log P_2}$ corresponding to the collision probabilities in Table 3.1 ($d > 24$). Probabilities P_1 and P_2 are the collision probabilities corresponding to radii R and cR respectively.

is still far from the desired exponent of 0.25.

For the case when $d \leq 24$, the collision probabilities are summarized in Table 3.3. The method for computing the probabilities is as before, except for the generation of the point q. In this case, the vector $q - p$ is a random vector of *fixed* length. The resulting values of ρ are summarized in Table 3.4.

3.4 Near-linear Space

In this section we present a data structure for the c-NN problem achieving near-linear space and $dn^{O(1/c^2)}$ query time. Our goal is to prove the following theorem.

Theorem 3.4.1. *There exists an algorithm solving the c-NN problem in the d-dimensional Euclidean space that achieves $dn^{O(1/c^2)}$ query time and $\tilde{O}(nd)$ space and preprocessing time.*

From a high-level, we obtain this data structure by plugging our new LSH function into the algorithm of Panigrahy [Pan06]. Unlike the standard LSH scheme of [IM98] (described in Section 3.1), which uses n^ρ independent hash tables, Panigrahy's algorithm uses only *one* such hash table to store the data set D. The hash table is then probed by hashing not

Radius	Est. collision probability	Confidence interval
0.7	0.0744600	[0.0744548, 0.0744653]
0.8	0.0424745	[0.0424705, 0.0424786]
0.9	0.0223114	[0.0223084, 0.0223144]
1.0	0.0107606	[0.0107585, 0.0107627]
1.1	0.0046653	[0.0046639, 0.0046667]
1.2	0.0017847	[0.0017838, 0.0017856]
1.3	0.0005885	[0.0005880, 0.0005890]
1.4	0.0001602	[0.0001599, 0.0001605]
1.5	0.0000338	[0.0000337, 0.0000340]
1.6	0.0000073	[0.0000072, 0.0000074]
1.7	0.0000009	[0.0000008, 0.0000010]
1.8	0.0000000	[0.0000000, 0.0000001]

Table 3.3: Probabilities of collision of two points, for $d \leq 24$, under the hash function described in Figure 3-6 with bounded-distance Leech decoder. The values were obtained through Monte-Carlo simulation for 10^7 trials. Confidence interval corresponds to 95% accuracy.

c	Max likelihood of ρ	Conservative ρ	Radius R
1.5	0.4402	0.4405	1
2.0	0.2671	0.2674	0.8

Table 3.4: The values of $\rho = \frac{\log P_1}{\log P_2}$ corresponding to the collision probabilities in Table 3.1 ($d \leq 24$). Probabilities P_1 and P_2 are the collision probabilities corresponding to radii R and cR respectively.

just the query point q (as in Section 3.1) but by hashing several points chosen randomly from the neighborhood of q. The intuition behind this approach is as follows. Let $p^* \in D$ be a point within distance 1 from q, i.e., p^* is a 1-NN. If a random LSH function causes collision between p^* and q with probability $1/n^\rho$, then it is plausible that, with constant probability[5], a random hash function causes collision between p^* and a "non-negligible" (say, $\approx 1/n^\rho$) fraction of the points in the unit ball around q. In such case, it would suffice to pick $\approx n^\rho$ random points from the unit ball around q and probe them into the hash table. Then we expect at least one of these random points to collide with p^*. Furthermore, any point from the unit ball around q cannot get too close to a "far" point p from the dataset by triangle inequality. Indeed, [Pan06] shows that, for the LSH family from [DIIM04], the above condition is satisfied (with a mild loss in parameters).

Our proof follows the above general intuition. However, in our case, converting this intuition into a formal proof is more technical. One of the reasons is due to the fact that our new LSH functions are more complex than the ones from [DIIM04], requiring us to extend [Pan06]'s framework to a more general setting.

Our algorithm construction and proof of correctness is structured as follows. First, we give an algorithm $\mathcal{A}_G$ that is a generalization of the [Pan06]'s algorithm summarized above,

[5]In the actual proof, the probability is $1/\log^{O(1)} n$.

and which works for a broad class of LSH families. Then, we discuss the parameters of this general algorithm, and prove its correctness. Finally, we show how to combine this general algorithm with our new LSH family from Section 3.2.1, and prove that the resulting query time is $dn^{O(1/c^2)}$.

Our generalized algorithm. Our algorithm $\mathcal{A}_G$ is an extension of the Panigrahy's algorithm from [Pan06], with the following important differences. For one, our probes are not chosen directly from the ball around the query q. Rather, the probes are chosen in the intermediary $\mathbb{R}^t$ space, between the random projection and the discretization step. This modification helps on several fronts. First of all, it significantly simplifies the analysis of the equivalent of the statement that "large fraction of the ball around q intersects with the bucket of p^*". Second, the query algorithm becomes slightly faster since we bypass the dimensionality reduction step in the hash function computation for the random probes.

The second difference from the [Pan06]'s algorithm is that our generalized algorithm supports LSH hash functions with arbitrary discretization stages. In fact, we only require the following condition on the hashing family: the family $\mathcal{H}$ is a composition of two component families of hashing functions, $\mathcal{F}$ and $\tilde{\mathcal{H}}$. We choose a random $h \in \mathcal{H}$ by setting it to be $h(x) = \tilde{h}(f(x))$, where f and $\tilde{h}$ are chosen independently at random from families $\mathcal{F}$ and $\tilde{\mathcal{H}}$ respectively. We assume that the functions $f \in \mathcal{F}$ map $\mathbb{R}^d$ into $\mathbb{R}^t$, and the functions $\tilde{h} \in \tilde{\mathcal{H}}$ map $\mathbb{R}^t$ into Ω, a discrete (potentially infinite) set. For example, this framework includes both the LSH function from [DIIM04] as well as our new LSH function from Section 3.2.1. Namely, the LSH function from Section 3.2.1 is $h(p) = \tilde{h}(f(p))$, where $f \in \mathcal{F}$ is the dimensionality reduction step and $\tilde{h} \in \tilde{\mathcal{H}}$ is the "ideal" LSH function. Thus, in the case of our LSH function, $\mathcal{F}$ is the set of dimensionality reduction functions $f : \mathbb{R}^d \to \mathbb{R}^t$, and $\tilde{\mathcal{H}}$ is the set of "ideal" LSH functions $\tilde{h} : \mathbb{R}^t \to \Omega$.

We present our algorithm $\mathcal{A}_G$ in Fig. 3-7. The algorithm $\mathcal{A}_G$ is parametrized by values k, ρ, and a probability distribution density function $\mathrm{P}^*(x)$.

Preprocessing:

1. Randomly pick a k-tuple $F = (f_1, f_2, \dots f_k)$ of functions $f_i \leftarrow \mathcal{F}$.
2. Randomly pick a k-tuple $\tilde{H} = (\tilde{h}_1, \tilde{h}_2, \dots \tilde{h}_k)$ of functions $\tilde{h}_i \leftarrow \tilde{\mathcal{H}}$.
3. For all $p \in D$, store $\tilde{H}(F(p)) = \left(\tilde{h}_1(f_1(p)), \tilde{h}_2(f_2(p)), \dots \tilde{h}_k(f_k(p))\right)$ in a single hash table.

Query q**:**

1. For each $i = 1, 2, \dots, O(n^\rho)$:
    2. Randomly pick $V_i = (v_{i,1}, v_{i,2}, \dots v_{i,k})$, where each $v_{i,j} \in \mathbb{R}^t$ is distributed according to the distribution P^*.
    3. Examine the points in the bucket $\tilde{H}(F(q) + V_i) = \left(\tilde{h}_1(f_1(q) + v_{i,1}), \dots \tilde{h}_k(f_k(q) + v_{i,k})\right)$. If the bucket contains some point p^* at distance $\|q - p^*\| \le c$, return p^* and stop.

Figure 3-7: Generalized algorithm $\mathcal{A}_G$ for obtaining a near-linear space LSH algorithm. The algorithm is parametrized by values k, ρ and a probability distribution P^*.

Parameters of the general algorithm $\mathcal{A}_G$. We now discuss the choice of the parameters k, ρ, and distribution $\mathrm{P}^*(x)$, which govern the correctness and efficiency of $\mathcal{A}_G$. We will give precise requirements later, in Lemma 3.4.4.

We start from $\mathrm{P}^*(x)$. As defined in preliminaries, $\mathrm{P}_{f(q)}$ will denote the probability distribution of $f(q)$ when $f \in \mathcal{F}$. Our ideal goal is to define $\mathrm{P}^* = \mathrm{P}_{f(p^*)-f(q)}$ where $p^* \in D$ is at distance exactly 1 from q, i.e., it is a 1-NN from the dataset. However, note that we do not know p^* a priori, and thus, in general, we do not know the distribution $\mathrm{P}_{f(p^*)-f(q)}$. Fortunately, we use a family $\mathcal{F}$ that automatically guarantees that $\mathrm{P}_{f(p^*)-f(q)}$ is the *exact same* distribution for all $p^* \in \bar{B}(q, 1)$ — abusing notation, we call this the "spherical symmetry" property of P^*. One minor further complication is that a desired 1-NN p^* may be closer than at distance exactly 1. In this case we guess the distance $\|q - p^*\|$ up to within a factor of, say, $1+\epsilon$ by trying all the radii $(1+\epsilon)^{-j}$ for $1 \leq j \leq O(\epsilon^{-1} \log\log n)$. Furthermore, we show that having only an approximation to $\|q - p^*\|$ suffices, as long as we set $\mathrm{P}^*(x)$ to be a distribution that is statistically close to $\mathrm{P}_{f(p^*)-f(q)}(x)$ when p^* is such that $\|p^* - q\| \in [1-\epsilon, 1]$.

The remaining parameters k and ρ are set as in [Pan06]. Specifically, we set $k = \frac{\log n}{\log 1/P_2}$, where P_2 is the probability of collision of two points at distance at least c, under the a random hash function $h \in \mathcal{H}$.

To specify the value of ρ, we need the following definition:

Definition 3.4.2. *Consider any hash function $\tilde{h} \in \tilde{\mathcal{H}}$, $\tilde{h} : \mathbb{R}^t \to U$. For a probability distribution function $\mathrm{P}^* : \mathbb{R}^t \to [0, 1]$, the* entropy *of $\tilde{h}$ with respect to distribution P^* is defined to be the entropy of the discrete random variable $\tilde{h}(v)$, where v is distributed according to the distribution $\mathrm{P}^*(v)$. We denote this entropy by $I(\tilde{h}(v) \mid \tilde{h})$.*

We set $\rho = \frac{M}{\log 1/P_2}$, where $M = \mathbb{E}_{\tilde{h}\in\tilde{\mathcal{H}}}\left[I(\tilde{h}(v) \mid \tilde{h})\right]$ is the expected entropy of a random function $\tilde{h} \in \tilde{\mathcal{H}}$.

Analysis of the general algorithm. We now proceed to proving the correctness and performance guarantees of the general algorithm $\mathcal{A}_G$, only assuming some properties of the LSH family (namely, via parameters P^*, k, ρ). We thus state a general lemma so that designers of future LSH families do not have to reproduce the entire proof. Later in this section, we will show how our new LSH family from Section 3.2.1 satisfies the required properties, and thus conclude Theorem 3.4.1.

Before stating the main lemma for the general algorithm $\mathcal{A}_G$, we need to define some natural technical condition.

Definition 3.4.3 (Translation invariance)**.** *Consider a family $\tilde{\mathcal{H}}$ of functions $\tilde{h} : \mathbb{R}^t \to U$. A function $\tilde{h} \in \tilde{H}$ induces an* equivalence function *$\psi_{\tilde{h}} : \mathbb{R}^t \times \mathbb{R}^t \to \{0, 1\}$, where $\psi_{\tilde{h}}(x, y) \triangleq 1$ if $\tilde{h}(x) = \tilde{h}(y)$ and $\psi_{\tilde{h}}(x, y) \triangleq 0$ if $\tilde{h}(x) \neq \tilde{h}(y)$. We denote by $\{\psi_{\tilde{h}}(x, y)\}_{\tilde{h}\in\tilde{\mathcal{H}}}$ the distribution of functions $\psi_{\tilde{h}}$ for $\tilde{h}$ chosen randomly from $\tilde{\mathcal{H}}$.*

We call $\tilde{\mathcal{H}}$ translation invariant *if, for any translation $s \in \mathbb{R}^t$, the distribution of $\psi_{\tilde{h}}(x, y)_{\tilde{h}\in\mathcal{H}}$ is the same as the distribution of $\{\psi_{\tilde{h}}(x + s, y + s)\}_{\tilde{h}\in\tilde{\mathcal{H}}}$.*

One can view the equivalence function $\psi_{\tilde{h}}$ as defining the equivalence relation on points in $\mathbb{R}^t$, two points being equivalent iff they fall into the same bucket under $\tilde{h}$. Translation invariance of $\tilde{\mathcal{H}}$ then says that even if a random function $\tilde{h}$ is translated, we do not change the distribution of equivalence relations on points induced by $\tilde{h}$.

The main correctness lemma follows.

Lemma 3.4.4. *Consider two families of functions $\mathcal{F}$ and $\tilde{\mathcal{H}}$, where a function $f \in \mathcal{F}$ acts from $\mathbb{R}^d$ to $\mathbb{R}^t$ and a function $\tilde{h} \in \tilde{\mathcal{H}}$ acts from $\mathbb{R}^t$ to U. Set $k = \frac{\log n}{\log 1/P_2}$ and $\rho = \frac{M}{\log 1/P_2}$, where $M = \mathbb{E}_{\tilde{h} \in \tilde{\mathcal{H}}}\left[I(\tilde{h}(v) \mid \tilde{h})\right]$. Suppose also the following properties hold:*

1. *The family $\tilde{\mathcal{H}}$ is translation invariant.*

2. P^* *is "spherically symmetric" in the sense that it is equal to $\mathrm{P}_{f(p^*)-f(q)}$ for each p^* at distance 1 from q. Also, there exists some $\epsilon > 0$ such that, for every p^* at distance $\|p^* - q\| \in [1-\epsilon, 1]$, the statistical distance between P^* and $\mathrm{P}_{f(p^*)-f(q)}$ is at most $O(1/(k \log^2 n))$.*

3. *For any $q, p^n, p^f \in \mathbb{R}^d$ such that $\|q - p^n\| = 1$ and $\|q - p^f\| > c$, there exists some point p^m with $\|q - p^m\| > c$, such that the distribution of $(f_1(p^n) - f_1(q)) - \big(f_2(p^f) - f_2(q)\big)$, for $f_1, f_2 \in \mathcal{F}$, has a pdf equal to $\mathrm{P}_{f(p^m)-f(q)}$.*

Then, the algorithm $\mathcal{A}_G$ has the following guarantees:

- *if there exists some $p^* \in D$ such that $\|q - p^*\| \in [1-\epsilon, 1]$, the algorithm $\mathcal{A}_{\mathcal{G}}$ will report a c-NN, with probability at least $\Omega(1/\log^2 n)$; and*
- *the expected number of points examined by the algorithm before stopping is $O(n^\rho)$.*

The proof of this lemma appears in Section 3.4.1.

A few remarks are in place. First, we note that even though the lemma guarantees a probability of success of only $\Omega(1/\log^2 n)$, we can boost this probability to a constant by creating $O(\log^2 n)$ parallel copies of the algorithm. In addition, we can eliminate the assumption that the near neighbor p^* is at distance $\|q - p^*\| \in [1-\epsilon, 1]$ by guessing the value $\|q - p^*\|$ to within a factor of $1+\epsilon$, and applying the algorithm $\mathcal{A}_G$ for each guess. Namely, we can rescale everything by $(1+\epsilon)^{-j}$ for each $j \in \{0, 1, \ldots \log_{1+\epsilon} \log n\}$ and run the algorithm (note that, for an $O(\log n)$ approximation, we immediately have a near-linear space algorithm by the basic algorithm from Section 3.1.1). Both of these modifications introduce the space and time overhead of at most $O(\frac{1}{\epsilon} \log^3 n)$, which is subsumed by the final $dn^{O(1/c^2)}$ query time bound.

Algorithm $\mathcal{A}_G$ with the new LSH family from Section 3.2.1. As mentioned before, our new LSH family fits precisely in the framework of the general algorithm $\mathcal{A}_G$. Specifically, we identify the family $\mathcal{F}$ with the family of random dimensionality reductions from $\mathbb{R}^d$ to $\mathbb{R}^t$, and we identify $\tilde{\mathcal{H}}$ with the family of "ideal" LSH functions in $\mathbb{R}^t$ described in Section 3.2.1. Thus, our hope would be to apply the Lemma 3.4.4 from above for our families $\mathcal{F}, \tilde{\mathcal{H}}$, and conclude that $\mathcal{A}_G$ has query time of $n^{O(1/c^2)}$. Indeed, we show below how to apply Lemma 3.4.4 for our LSH family.

We need to prove two steps. First, we need to show that we satisfy the conditions of the Lemma 3.4.4. Second, we need to prove that $\rho = O(1/c^2)$, thus concluding, by Lemma 3.4.4, that the query time of the resulting algorithm is $n^{O(1/c^2)}$.

We prove below that we satisfy all three conditions of Lemma 3.4.4.

1. We need to prove that the "ideal" LSH functions $\tilde{\mathcal{H}}$ is translation invariant. Translation invariance requires that, for any $s \in \mathbb{R}^t$, the distribution of $\{\psi_{\tilde{h}(z)}(x,y)\}_{\tilde{h}(z) \in \tilde{\mathcal{H}}}$ is the same as $\{\psi_{\tilde{h}(z)}(x+s, y+s)\}_{\tilde{h}(z) \in \tilde{\mathcal{H}}}$. Indeed, note that $\psi_{\tilde{h}(z)}(x+s, y+s) = \psi_{\tilde{h}(z+s)}(x,y)$.

Thus, the distribution $\{\psi_{\tilde{h}(z)}(x+s, y+s)\}_{\tilde{h}(z)\in\tilde{\mathcal{H}}}$ is the same as $\{\psi_{\tilde{h}(z+s)}(x,y)\}_{\tilde{h}(z)\in\tilde{\mathcal{H}}}$. Furthermore, the distribution $\{\psi_{\tilde{h}(z+s)}(x,y)\}_{\tilde{h}(z)\in\tilde{\mathcal{H}}}$ is the same as $\{\psi_{\tilde{h}(z)}(x,y)\}_{\tilde{h}(z)\in\tilde{\mathcal{H}}}$ since our "ideal" LSH functions $h(z)$ are defined by the positions of the grids G_u^t which themselves are shifted by a random vector.

2. The second property requires that $\mathrm{P}^*(x)$ is statistically close to $\mathrm{P}_{f(p^*)-f(q)}(x)$, for each p^* such that $\|q-p^*\| \in [1-\epsilon, 1]$. Since $f \in \mathcal{F}$ is a random dimensionality reduction, we set P^* is set to be a t-dimensional Gaussian distribution, i.e., $\mathrm{P}^* \triangleq N(0, 1/\sqrt{t})^t$.

   We now prove the statistical distance property for $\epsilon = O(1/kt\log^2 n)$. Notice that, for p^* such that $\|q-p^*\| = \delta \in [1-\epsilon, 1]$, distribution $\mathrm{P}_{f(p^*)-f(q)}$ is equal to $N(0, \delta/\sqrt{t})^t$.

   We bound the statistical distance $\mathrm{TV}(\mathrm{P}_{f(p^*)-f(q)}, \mathrm{P}^*)$ as follows:

$$\begin{aligned}
\mathrm{TV}(\mathrm{P}_{f(p^*)-f(q)}, \mathrm{P}^*) &= \int\limits_{\mathbb{R}^t} \max\left\{0, \prod_i \frac{\sqrt{t}}{\sqrt{2\pi}\delta} e^{-(x_i\sqrt{t}/\delta)^2/2} - \prod_i \frac{\sqrt{t}}{\sqrt{2\pi}} e^{-(x_i\sqrt{t})^2/2}\right\} \prod_i dx_i \\
&= \int\limits_{\mathbb{R}^t} \prod_i \frac{\sqrt{t}}{\sqrt{2\pi}} e^{-(x_i\sqrt{t})^2/2} \max\left\{0, \prod_i \frac{1}{\delta} \exp\left[x_i^2 t(1-\tfrac{1}{\delta^2})/2\right] - 1\right\} \prod_i dx_i \\
&\le \int\limits_{\mathbb{R}^t} \prod_i \frac{\sqrt{t}}{\sqrt{2\pi}} e^{-(x_i\sqrt{t})^2/2} \max\left\{0, \frac{1}{\delta^t}\cdot 1 - 1\right\} \prod_i dx_i \\
&= \frac{1}{\delta^t} - 1 \\
&\le O(1/k\log^2 n).
\end{aligned}$$

3. Fix some points q, p^n, and $p^f \in \mathbb{R}^d$ satisfying $\|q-p^n\| = 1$ and $\|q-p^f\| \ge c$. We need to show that the distribution of $(f_1(p^n)-f_1(q)) - \left(f_2(p^f)-f_2(q)\right)$ for $f_1, f_2 \in \mathcal{F}$ has a pdf equal to $\mathrm{P}_{f(p^m)-f(q)}$ for some $p^m \notin B(q,c)$. Indeed, if we let $\delta = \|q-p^f\|$, then $f_1(p^n)-f_1(q)$ is distributed as $(N(0,1/\sqrt{t}))^t$, and $f_2(p^f)-f_2(q)$ is distributed as $(N(0,\delta/\sqrt{t}))^t$. Thus, $f_1(p^n)-f_1(q)-(f_2(p^f)-f_2(q))$ is distributed as $(N(0,\sqrt{1+\delta^2}/\sqrt{t}))^t$ by Fact 2.4.5. Now, choose p^m to be some fixed point at distance $\sqrt{1+\delta^2} > \delta > c$ from q, and note that $f(p^m)-f(q)$ is also distributed as $(N(0,\sqrt{1+\delta^2}/\sqrt{t}))^t$.

Complexity analysis. Finally, we analyze the complexity of the algorithm. First, we need to show that $\rho = O(1/c^2)$. We set $w = \sqrt{t}/2$. In that case, we can show that $P_2 \le e^{-\Omega(c^2)}$. Indeed, using the calculation from Eqn. (3.3) with $\Delta' \ge c(1-t^{-1/4})$ and $f = \exp\left[-\Omega(\sqrt{t})\right]$, we have that

$$P_2 \le 2\exp\left[-\tfrac{t}{2}(c(1-t^{-1/4})/2w)^2\right] + f \le 3\exp\left[-c^2/3\right],$$

and then $\log 1/P_2 \ge \Omega(c^2)$.

It just remains to show that $M = \mathbb{E}_{\tilde{h}\in\tilde{\mathcal{H}}}\left[I(\tilde{h}(v) \mid \tilde{h})\right] = O(1)$. Then we can conclude that $\rho = \frac{M}{\log 1/P_2} \le K/c^2$ for some constant K. We will prove the desired bound on M in the following lemma.

Lemma 3.4.5. *Let $\tilde{h} \in \tilde{\mathcal{H}}$. Let w be such that $1 \le w \le \sqrt{t}$. Then $\mathbb{E}_{\tilde{h}}\left[I(\tilde{h}(v) \mid \tilde{h})\right] \le O\left(\left(\frac{\sqrt{t}}{w}\right)^3 \exp\left[\frac{t}{2w^2}\right]\right)$, where v is chosen according to a t-dimensional Gaussian distribution $(N(0, 1/\sqrt{t}))^t$.*

To finish the proof of Theorem 3.4.1, it remains to prove Lemmas 3.4.4 and 3.4.5, which we do in the following two sections.

3.4.1 Correctness of the general algorithm: proof of Lemma 3.4.4

The proof is organized in four parts. First, we show that the behavior of algorithm $\mathcal{A}_G$ is equivalent to the behavior of a slightly different algorithm $\mathcal{B}_G$, which combines the preprocessing and the query stages and is easier to analyze. Second, we prove that, if $\mathrm{P}^* = \mathrm{P}_{f(p^*)-f(q)}$ where p^* is at distance exactly 1 from q, the algorithm $\mathcal{B}_G$ returns correct answer with a probability at least $\Omega(1/\log^2 n)$. Third, we prove that the success probability does not decrease by too much if P^* is only statistically close to $\mathrm{P}_{f(p^*)-f(q)}$. Finally, we prove that $\mathcal{B}_G$ examines at most $O(n^\rho)$ points in expectation.

We give the modified algorithm $\mathcal{B}_G$ in Fig. 3-8, which we show to be equivalent to the algorithm $\mathcal{A}_G$.

1. Randomly pick a k-tuple $F' = (f'_1, f'_2, \dots f'_k)$ of functions $f'_i \leftarrow \mathcal{F}$.
2. Randomly pick a k-tuple $\tilde{H}' = (\tilde{h}'_1, \tilde{h}'_2, \dots \tilde{h}'_k)$ of functions $\tilde{h}'_i \leftarrow \tilde{\mathcal{H}}$.
3. Let $\Psi_{\tilde{H}'} = (\psi_{h'_1}, \psi_{h'_2}, \dots \psi_{h'_k})$ where $h'_i = \tilde{h}'_i \circ f'_i$.
4. For each $i = 1, 2, \dots, O(n^\rho)$:
   5. Randomly pick $V'_i = (v'_{i,1}, v'_{i,2}, \dots v'_{i,k})$, where each $v'_{i,j} \in \mathbb{R}^t$ is distributed according to the distribution P^*.
   6. Examine the points $p \in D$ such that $\Psi_{\tilde{H}'}(F'(p) - F'(q), V'_i) = 1$. If some examined point p^* is at distance $\|q - p^*\| \le c$, return p^* and stop.

Figure 3-8: The modified algorithm $\mathcal{B}_G$, equivalent to the algorithm $\mathcal{A}_G$ from Fig. 3-7, assuming the conditions of Lemma 3.4.4.

Equivalence of $\mathcal{A}_G$ and $\mathcal{B}_G$. We show that the algorithms $\mathcal{A}_G$ and $\mathcal{B}_G$ produce exactly the same answers if: $F = F'$, $V_i = V'_i$ for all probes i, and $\Psi_{\tilde{H}}(F(p), F(q)+V_i) = \Psi_{\tilde{H}'}(F(p) - F(q), V_i)$ for all q, p, F and i. To understand these conditions, note that the algorithm $\mathcal{A}_G$, in query step 3, examines the points $p \in D$ such that $\Psi_{\tilde{H}}(F(p), F(q) + V) = 1$. Thus, if $\Psi_{\tilde{H}}(F(p), F(q)+V) = \Psi_{\tilde{H}'}(F(p) - F(q), V)$, then $\mathcal{A}_G$ and $\mathcal{B}_G$ examine exactly the same set of points.

Now recall that the family $\tilde{\mathcal{H}}$ is translation invariant. Therefore the tuples $\langle F, \{V_i\}, \Psi_{\tilde{H}}(x, y)\rangle$ and $\langle F', \{V'_i\}_i, \Psi_{\tilde{H}'}(x - F'(q), y - F'(q))\rangle$ have the same distributions. Thus, $\mathcal{A}_G$ and $\mathcal{B}_G$ have exactly the same behavior, and it suffices to analyze the algorithm $\mathcal{B}_G$ only.

Before continuing with the analysis of the algorithm $\mathcal{B}_G$, we summarize our notation in Table 3.5 for the ease of reference.

Success probability of $\mathcal{B}_G$. When discussing the "success probability of $\mathcal{B}_G$", we mean the following condition. Suppose there is some $p^* \in D$ such that $\|q - p^*\| \in [1 - \epsilon, 1]$. We

Notation	Definition
$F' = (f'_1, \dots f'_k)$	a hash function chosen from the family $\mathcal{F}^k$
$\tilde{H}' = (h'_1, \dots h'_k)$	a hash function chosen from the family $\tilde{\mathcal{H}}^k$
$\Psi_{\tilde{H}'}(x, y)$	the equivalence function of $\tilde{H}'$: equal to 1 if $\tilde{H}'(x) = \tilde{H}'(y)$, and 0 otherwise
$V'_i = (v'_{i,1}, \dots v'_{i,k})$	the i^{th} probe, chosen from the distribution $(\mathrm{P}^*)^k$
q	the query point
p^*	the desired near neighbor, satisfying $\|q - p^*\| \in [1-\epsilon, 1]$
p^n	a point such that $\|q - p^n\| = 1$ (see precondition 3)
p^f	a point from D such that $\|q - p^f\| > c$ (see precondition 3)
p^m	a point at distance at least c from q (see precondition 3)
$M = I(\tilde{H}'(V') \mid \tilde{H}')$	the entropy of the partition induced by $\tilde{H}'(V')$, for $V' \leftarrow (\mathrm{P}^*)^k$, for fixed $\tilde{H}'$
$\mathrm{P}_{f(p)-f(q)}$	the distribution of $f(p) - f(q)$ when $f \leftarrow \mathcal{F}$

Table 3.5: Notation used in the proof of Lemma 3.4.4.

say $\mathcal{B}_G$ *succeeds* if $\Psi_{\tilde{H}'}(F'(p^*) - F'(q), V'_i) = 1$ for at least one probe V'_i — in this case, the algorithm guarantees to return some c-NN. Note that if no such $p^* \in D$ exists, we do not require anything from $\mathcal{B}_G$, except that $\mathcal{B}_G$ examines at most $O(n^\rho)$ points in expectation. Thus, for discussing the success probability of $\mathcal{B}_G$, we assume that $p^* \in D$.

We now prove that the algorithm $\mathcal{B}_G$ succeeds with probability at least $\Omega(1/\log^2 n)$. At the moment, we assume that $\mathrm{P}^* = \mathrm{P}_{f(p^*)-f(q)}$, where $p^* \in D$ is at distance $\|q-p^*\| \in [1-\epsilon, 1]$. We will remove this assumption later. The claim follows from Lemma 2 in [Pan06], which states the following:

Lemma 3.4.6 ([Pan06])**.** *Suppose we are given a discrete probability space Ω with entropy I, and a "reference" sample r^* drawn from Ω. If we draw $\gamma = O(2^I)$ additional samples (probes) $r_1, \dots r_\gamma$ from Ω, then $r^* = r_i$ for some $i \in [\gamma]$ with probability at least $\Omega(1/I)$.*

We apply this lemma in the following setting: the sample space is the partition induced by the function $\tilde{H}'$, the reference sample is $r^* = \tilde{H}'(F'(p^*) - F'(q))$, whereas the additional samples are $r_i = \tilde{H}'(V'_i)$, for $i \in \{1, \dots O(n^\rho)\}$. Furthermore, if $r^* = r_i$ for some probe $i \in \{1, \dots O(n^\rho)\}$, then $\tilde{H}'(F'(p^*)-F'(q)) = \tilde{H}'(V'_i)$, implying that $\Psi_{\tilde{H}'}(F'(p^*)-F'(q), V'_i) = 1$, i.e., that the algorithm $\mathcal{B}_G$ succeeds. Note that, in this setting, the entropy is $I = I(\tilde{H}'(V') \mid \tilde{H}')$, i.e., the entropy of the variable $\tilde{H}'(V')$ when V' is drawn from $(\mathrm{P}^*)^k = \mathrm{P}^* \times \mathrm{P}^* \times \dots \mathrm{P}^*$, assuming a fixed $\tilde{H}'$.

We note that the lemma has a relatively natural intuition behind it. Specifically, if the entropy is very small — zero in the extreme — then there is only one bucket. This bucket contains the desired point p^*, and thus we need only a few probes to find the bucket of p^*. On the other hand, if the entropy is large, as given by uniform distribution on $k = 2^I$ buckets, then we would need $O(k)$ probes to hit one particular bucket, which contains the desired p^*.

To use the Lemma 3.4.6 in our context, we just need to show that $\mathcal{B}_G$ makes at least γ probes, i.e., $\gamma = O(2^I) \leq O(n^\rho)$, and then we can use the lemma to obtain a lower bound on the success probability of $\mathcal{B}_G$. While we cannot prove that $\gamma \leq O(n^\rho)$ always, we show it happens with good enough probability over the choice of $\tilde{H}'$. To prove this, we compute an upper bound for $I = I(\tilde{H}'(V') \mid \tilde{H}')$. By Markov's inequality, $I(\tilde{H}'(V') \mid \tilde{H}') \leq (1 + 1/\log n)\mathbb{E}_{\tilde{H}' \in \tilde{\mathcal{H}}}\left[I(\tilde{H}'(V') \mid \tilde{H}')\right]$ with probability at least $1/\log n$. Furthermore, note that

$$\mathbb{E}_{\tilde{H}'}\left[I(\tilde{H}'(V') \mid \tilde{H}')\right] \le \mathbb{E}_{\tilde{H}'}\left[\sum_{j=1}^{k} I(\tilde{h}'_j(v'_j) \mid \tilde{h}'_j)\right] = k \cdot \mathbb{E}_{\tilde{h}\in\tilde{\mathcal{H}}}\left[I(\tilde{h}(v) \mid \tilde{h})\right],$$

since each $\mathbb{E}_{\tilde{h}'_j}\left[I(\tilde{h}'_j(v'_j) \mid \tilde{h}'_j)\right]$ is the expected entropy of a function $\tilde{h} \in \tilde{\mathcal{H}}$ with respect to P^*. Thus, with probability at least $1/\log n$, we have that $I \le (1+1/\log n)\cdot k\mathbb{E}_{\tilde{h}\in\tilde{\mathcal{H}}}\left[I(\tilde{h}(v) \mid \tilde{h})\right] = (1+1/\log n)kM$, where $M = \mathbb{E}_{\tilde{h}\in\tilde{\mathcal{H}}}\left[I(\tilde{h}(v) \mid \tilde{h})\right]$. Replacing $k = \frac{\log n}{\log 1/p_2}$, we obtain

$$\gamma = O(2^I) = O\left(2^{(1+1/\log n)\cdot\frac{\log n}{\log 1/P_2}M}\right) = O(n^{\frac{M}{\log 1/P_2}}) = O(n^\rho).$$

Thus, with probability at least $1/\log n$, the algorithm chooses a tuple $\tilde{H}'$ for which $\gamma \le O(n^\rho)$. Conditioned on such a choice of $\tilde{H}'$, by Lemma 3.4.6, with probability $\Omega(1/I) = \Omega(1/\log n)$, at least one of the $O(n^\rho)$ probes r_i equals to r^*. The total success probability of $\mathcal{B}_G$ becomes $1/\log n \cdot \Omega(1/\log n) = \Omega(1/\log^2 n)$.

Removing the assumption on distribution P^*. In the next step we remove the assumption that $\mathrm{P}^* = \mathrm{P}_{f(p^*)-f(q)}$, and show that the algorithm $\mathcal{B}_G$ maintains a success probability of $\Omega(1/\log^2 n)$ even if we assume only that $\mathrm{TV}(\mathrm{P}^*, \mathrm{P}_{f(p^*)-f(q)}) \le O(1/k\log^2 n)$. This follows from the standard fact on statistical distance, namely Fact 2.4.1. Specifically, consider a new algorithm, $\mathcal{B}'_G$, which differs from $\mathcal{B}_G$ in step 6, where it checks only whether $\Psi_{\tilde{H}'}(F'(p^*) - F'(q), V'_i) = 1$ for p^*, instead of doing that for all points $p \in D$. Note that this modification does not decrease the success probability of the algorithm. Then, we can view $\mathcal{B}'_G$ as a randomized algorithm on the input $x = F'(p^*) - F'(q)$, with $\tilde{H}'$ and V'_i being the random choices of $\mathcal{B}'_G$. $\mathcal{B}'_G$'s input $x = F'(p^*) - F'(q)$ is drawn from distribution $(\mathrm{P}_{f(p^*)-f(q)})^k$. As we have shown above, if we draw the input x from the distribution $(\mathrm{P}_{f(p^*)-f(q)})^k$, the success probability of $\mathcal{B}'_G$ is at least $\Omega(1/\log^2 n)$. Thus, by Fact 2.4.1, if x is drawn from P^* instead, then the success probability of $\mathcal{B}'_G(x)$ is at least $\Omega(1/\log^2 n) - \mathrm{TV}((\mathrm{P}^*)^k, (\mathrm{P}_{f(p^*)-f(q)})^k)$. It just remains to show that $\mathrm{TV}((\mathrm{P}^*)^k, (\mathrm{P}_{f(p^*)-f(q)})^k) < O(1/\log^2 n)$.

Indeed $\mathrm{TV}((\mathrm{P}^*)^k, (\mathrm{P}_{f(p^*)-f(q)})^k) \le O(1/\log^2 n)$ follows from condition (2) of the Lemma 3.4.4 and the triangle inequality on the statistical distance (see Fact 2.4.2):

$$\mathrm{TV}((\mathrm{P}^*)^k, (\mathrm{P}_{f(p^*)-f(q)})^k) \le k \cdot \mathrm{TV}(\mathrm{P}^*, \mathrm{P}_{f(p^*)-f(q)}) = O(1/\log^2 n).$$

Running time of $\mathcal{B}_G$. Finally, we need to argue that the algorithm $\mathcal{B}_G$ examines $O(n^\rho)$ points in expectation. Note that we need only to consider the "far" points $p^f \in D$ at distance $\|q - p^f\| > c$, since once we encounter a point at distance $\le c$, we return it. To prove an upper bound on the expected number of examined points, it is sufficient to prove that, for a probe V'_i, the condition $\Psi_{\tilde{H}'}(F'(p^f) - F'(q), V'_i) = 1$ is satisfied by only $O(1)$ far points $p^f \in D$ in expectation. Indeed, fix a far point p_f. Also, below we assume F_1, F_2 are drawn from $\mathcal{F}^k$, and we write $x \sim y$ to mean that random variables x and y have the same pdfs. We have the following equality of probability of distributions, using conditions (2) and (3) of Lemma 3.4.4:

$$\begin{array}{rcl}\Psi_{\tilde{H}'}(F'(p^f) - F'(q), V_i') & \sim & \Psi_{\tilde{H}'}(F'(p^f) - F'(q), F_1(p^n) - F_1(q)) \\ & \sim & \Psi_{\tilde{H}'}(F'(p^f) - F'(q) - (F_1(p^n) - F_1(q)) + F_1(p^n) - F_1(q), F_1(p^n) - F_1(q)) \\ & \sim & \Psi_{\tilde{H}'}(F_2(p^m) - F_2(q) + F_1(p^n) - F_1(q), F_1(p^n) - F_1(q)) \\ & \sim & \Psi_{\tilde{H}'}(F_2(p^m) - F_2(q) + F_1(p^n) - F_1(q), F_2(q) - F_2(q) + F_1(p^n) - F_1(q)).\end{array}$$

Furthermore, by translation invariance of $\tilde{\mathcal{H}}$, for $s = -F_2(q) + F_1(p^\dagger) - F_1(q)$, the distribution of the last expression is equal to the distribution of $\Psi_{\tilde{H}'}(F_2(p^m), F_2(q))$, which is the probability of collision of q and p^m under random LSH function $\tilde{H}' \circ F_2 \leftarrow \left(\tilde{\mathcal{H}} \circ \mathcal{F}\right)^k$, where p^m satisfies $\|q - p^m\| > c$. The probability of collision of p^m and q under $\tilde{H}' \circ F_2$ is at most P_2^k by the definition of P_2. Thus, the probability of examining a point p_f is at most $P_2^k = 1/n$ by the definition of k. Concluding, the expected number of examined points for one probe V_i' is at most $P_2^k \cdot 1/n = 1$. Over $O(n^\rho)$ probes, we examine an expected of $O(n^\rho)$ far points $p^f \in D$.

3.4.2 Entropy bound: proof of Lemma 3.4.5

Let $\tilde{h} \in \tilde{\mathcal{H}}$ be an "ideal" LSH function $\tilde{h}$ as described in Section 3.2.1, with some parameters t and w. Recall that the "ideal" LSH function $\tilde{h} \in \tilde{\mathcal{H}}$ is defined by grids G_u^t, $u \in [U_t]$, where $G_u^t = G^t + s_u$, $s_u \in [0, 4w]^t$, is a randomly shifted regular grid of balls of radius w.

We call $I(\tilde{h}) = I(\tilde{h}(v) \mid \tilde{h})$ the entropy of random variable $\tilde{h}(v)$, where v is distributed as a t-dimensional Gaussian distribution. One LSH function $\tilde{h}$ divides the probability space of v into some number of "cells," and the probability masses of these cells define the entropy $I(\tilde{h})$. We prove below that the expected entropy $\mathbb{E}_{\tilde{h}}\left[I(\tilde{h})\right] = O\left(\left(\frac{\sqrt{t}}{w}\right)^3 \exp\left[\frac{t}{2w^2}\right]\right)$.

We separate all balls defining $\tilde{h}$ into two groups: *far balls* with their center outside $B(0^t, 2w)$, and *close* balls with center inside $B(0^t, 2w)$. Note that most of the probability mass of v is inside $B(0^t, 2w)$, and thus we expect most of the entropy contribution to come from the close balls.

Indeed, we first show that the entropy contribution from far balls is exponentially small in t. For an integer $j \geq 2$, consider the balls with centers inside the spherical shell $R_j = B(0^t, jw + w) \setminus B(0^t, jw)$. There are less than $(j+1)^t$ such balls from a single grid, and thus there are at most $(j+1)^t \cdot U_t = 2^{O(t \log jt)}$ balls in R_j in total. Since each ball defines at most one cell, there are at most $N_j = 2^{O(t \log jt)}$ cells defined by these balls. But these balls contain probability mass smaller than $\Pr_v[\|v\| \geq (j-1)w]$, which is bounded by $\Pr_v[\|v\| \geq (j-1)w] \leq \pi_j = \exp\left[-\Omega(twj)\right]$, by Fact 2.3.3. Thus, the entropy contribution of the balls in the shell R_j is bounded by $\pi_j \cdot \ln N_j/\pi_j = \exp\left[-\Omega(tjw)\right]$. Finally, the contribution of the balls in all the shells $R_2, R_3, \ldots$ is bounded by $\sum_{j\geq 2} \exp\left[-\Omega(tjw)\right] \leq \exp\left[-\Omega(t)\right]$.

We now bound the entropy contribution of close balls. We denote this contribution by $I'(\tilde{h})$. Let B_1 be the first ball with center inside $B(0^t, 2w)$, B_2 the second ball inside $B(0^t, 2w)$, etc. Note that no two balls B_i and B_j, $i \neq j$, come from the same grid, and, thus, the positions of these balls are independent. Also, let f_i be the probability of v belonging to ball B_i and not to any of the previous balls (including the far balls). Let PF_i be the union of all far balls that appear before ball B_i. Then

$$f_i = \Pr_v[v \in B_i \setminus B_1 \setminus \cdots \setminus B_{i-1} \setminus PF_i\}].$$

Now $I'(\tilde{h})$ is the entropy of a sample space with probabilities $f_1, f_2, \ldots$, i.e., $I(\tilde{h}) = \sum_{i\geq 1} f_i \ln(1/f_i)$. Thus, the expectation of $I'(\tilde{h})$ is

$$\begin{aligned}\mathbb{E}_{\tilde{h}}\left[I'(\tilde{h})\right] &= \mathbb{E}_{\tilde{h}}\left[\sum_{i\geq 1} f_i \ln(1/f_i)\right] \\ &= \sum_{i\geq 1} \mathbb{E}_{\tilde{h}}\left[f_i \ln(1/f_i)\right]\end{aligned}$$

Our approach is to compute, for each i separately, the contribution of the ball i to the entropy, $\mathbb{E}_{\tilde{h}}\left[f_i \ln 1/f_i\right]$. The heart of the matter is the following bound.

Proposition 3.4.7. *The expected contribution of the ball B_i is*

$$\mathbb{E}_{\tilde{h}}\left[f_i \ln 1/f_i\right] \leq O\left(\left(\frac{\sqrt{t}}{w}\right)^3 \exp\left[\frac{t}{2w^2}\right]\right) \cdot 2^{-t} F_i \ln 2/F_i,$$

where $F_i = \exp\left[-(i-1)2^{-t}\right]$.

Proof. We start by noting that we can compute $\mathbb{E}_{\tilde{h}}\left[f_i \ln 1/f_i\right]$ as follows:

$$\begin{aligned}\mathbb{E}_{\tilde{h}}\left[f_i \ln 1/f_i\right] &= \mathbb{E}_{B_1,B_2,\ldots B_i,PF_i}\left[f_i \ln(1/f_i)\right] \\ &= \mathbb{E}_{B_i}\left[\mathbb{E}_{B_1,B_2,\ldots B_{i-1},PF_i}\left[f_i \ln(1/f_i)\right]\right] \\ &\leq \mathbb{E}_{B_i}\left[\mathbb{E}_{B_1,B_2,\ldots B_{i-1},PF_i}\left[f_i\right] \ln \frac{1}{\mathbb{E}_{B_1,B_2,\ldots B_{i-1},PF_i}\left[f_i\right]}\right] \qquad (3.5)\end{aligned}$$

where the last inequality follows from Fact 2.4.6.

In other words, we estimate $\mathbb{E}_{B_1,B_2,\ldots B_{i-1},PF_i}\left[f_i\right]$ assuming a fixed B_i, and then average it out over the random choice of the position of B_i. The quantity $\mathbb{E}_{B_1,B_2,\ldots B_{i-1},PF_i}\left[f_i\right]$ is composed of two factors. The first factor corresponds to the fraction of v's probability mass cut by B_i, i.e., $\Pr_v[v \in B_i]$. The second factor corresponds to the expected amount of v's probability mass left uncovered by the previous balls, $B_1, \ldots B_{i-1}, PF_i$, for which we show an upper bounded of $F_i = \exp\left[-(i-1)2^{-t}\right]$. We formalize this intuition below. We concentrate only on the balls B_i that are not too close to the origin, since, as we show later, the balls B_i close to the origin capture too little mass and have a negligible contribution. Formally, for balls B_i that are not too close to the origin, we show the following.

Fact 3.4.8. *Suppose B_i is at distance u from the origin, where $u \geq w$. Then $\mathbb{E}_{B_1,B_2,\ldots B_{i-1},PF_i}\left[f_i\right] \leq \exp\left[-t(u-w)^2/2\right] \cdot F_i$, where $F_i = \exp\left[-(i-1)2^{-t}\right]$.*

Proof. Denoting by $\chi[E]$ the indicator variable of an event E, we have

$$\begin{aligned}\mathbb{E}_{B_1,B_2,\ldots B_{i-1},PF_i}\left[f_i\right] &= \int_{x\in B_i} \mathrm{P}^*(x)\mathbb{E}_{B_1\ldots B_{i-1},PF_i}\left[\chi[x \notin B_1 \cup \cdots \cup B_{i-1} \cup PF_i]\right]dx \\ &\leq \int_{x\in B_i} \mathrm{P}^*(x)dx \cdot \left(1 - \frac{\mathrm{Vol}^t(w)}{\mathrm{Vol}^t(2w)}\right)^{i-1} \\ &= \int_{x\in B_i} \mathrm{P}^*(x)dx \cdot \left(1-2^{-t}\right)^{i-1} \\ &\leq F_i \cdot \int_{x\in B_i} \mathrm{P}^*(x)dx\end{aligned}$$

We now bound $\int_{x\in B_i} \mathrm{P}^*(x)dx$ when B_i's center is at a distance $u \geq w$ from the origin. First, rotate the t-dimensional space so that B_i is centered at $(u, 0, \ldots, 0)$. Let Y be a random variable drawn from P^*, i.e., the t-dimensional Gaussian distribution. If $x \in B_i$, then it must be the case that the first coordinate of x is greater than $u-w$. Thus we obtain:

$$\begin{aligned}
\int_{x\in B_i} \mathrm{P}^*(x)dx &\leq \int_{y=u-w}^{\infty} \frac{\sqrt{t}}{\sqrt{2\pi}} \exp\left[-y^2 \cdot t/2\right] dy \\
&= \exp\left[-t(u-w)^2/2\right] \cdot \int_{y=0}^{\infty} \frac{1}{\sqrt{2\pi}} \exp\left[-y^2/2 - y\sqrt{t}(u-w)\right] dy \\
&\leq \exp\left[-t(u-w)^2/2\right].
\end{aligned}$$

We thus conclude the proof of Fact 3.4.8. □

We state two more useful facts.

Fact 3.4.9. *For any* $0 \leq x \leq y \leq 1$, $x \ln 1/x \leq y \ln 2/y$.

Fact 3.4.10. *There exists some constant* C *such that for any* $y_0 \geq 1$, *we have*

$$\int_0^{\infty} \exp\left[-(y-y_0)^2/2\right] \cdot y^2 dy \leq C y_0^2.$$

Proof. Since $\frac{1}{\sqrt{2\pi}} e^{-x^2/2}$ is the Gaussian distribution with variance one, we have that $\int_{-\infty}^{\infty} \frac{1}{\sqrt{2\pi}} e^{-x^2/2} dx = \int_{-\infty}^{\infty} \frac{1}{\sqrt{2\pi}} e^{-x^2/2} x^2 dx = 1$. Thus,

$$\begin{aligned}
\int_0^{\infty} \exp\left[-(y-y_0)^2/2\right] \cdot y^2 dy &= \int_{-y_0}^{\infty} \exp\left[-y^2/2\right] \cdot (y+y_0)^2 dy \\
&\leq \int_{-y_0}^{y_0} \exp\left[-y^2/2\right] \cdot 4y_0^2 dy + \int_{y_0}^{\infty} \exp\left[-y^2/2\right] \cdot 4y^2 dy \\
&\leq 4\sqrt{2\pi} y_0^2 + 4\sqrt{2\pi} \\
&\leq 8\sqrt{2\pi} y_0^2.
\end{aligned}$$

□

We can now proceed to computing $\mathbb{E}_{\tilde{h}}\left[f_i \ln 1/f_i\right]$, which we do by integrating over all possible positions of B_i. By Eqn. (3.5), we have

$$\begin{aligned}
\mathbb{E}_{\tilde{h}}\left[f_i \ln \tfrac{1}{f_i}\right] &\leq \mathbb{E}_{B_i}\left[\mathbb{E}_{B_1,\ldots B_{i-1},PF_i}\left[f_i\right] \ln \frac{1}{\mathbb{E}_{B_1,\ldots B_{i-1},PF_i}\left[f_i\right]}\right] \\
&= \int_{u=0}^{2w} \frac{\mathrm{Sur}^t(u)}{\mathrm{Vol}^t(2w)} \cdot \mathbb{E}_{B_1,\ldots B_{i-1},PF_i}\left[f_i\right] \ln \frac{1}{\mathbb{E}_{B_1,\ldots B_{i-1},PF_i}\left[f_i\right]} \cdot du \\
&= L_i + \int_{u=w}^{2w} \frac{\mathrm{Sur}^t(u)}{\mathrm{Vol}^t(2w)} \cdot \mathbb{E}_{B_1,\ldots B_{i-1},PF_i}\left[f_i\right] \ln \frac{1}{\mathbb{E}_{B_1,\ldots B_{i-1},PF_i}\left[f_i\right]} \cdot du \qquad (3.6)
\end{aligned}$$

where $L_i = \int_{u=0}^{w} \frac{\mathrm{Sur}^t(u)}{\mathrm{Vol}^t(2w)} \cdot \mathbb{E}_{B_1,\ldots B_{i-1},PF_i}\left[f_i\right] \ln \frac{1}{\mathbb{E}_{B_1,\ldots B_{i-1},PF_i}\left[f_i\right]} \cdot du$ is the contribution of

B_i's close to the origin, and is a negligible term, as we show later. We have:

$$\begin{aligned}
\mathbb{E}_{\tilde{h}}\left[f_i \ln \tfrac{1}{f_i}\right] - L_i &\leq \int_{u=w}^{2w} \tfrac{\mathrm{Sur}^t(u)}{\mathrm{Vol}^t(2w)} \cdot \mathbb{E}_{B_1,\ldots B_{i-1},PF_i}[f_i] \ln \tfrac{1}{\mathbb{E}_{B_1,\ldots B_{i-1},PF_i}[f_i]} \cdot du & (3.7)\\
&\leq \int_{u=w}^{2w} \tfrac{t}{2w}(\tfrac{u}{2w})^{t-1} \cdot \exp\left[-(u-w)^2 \tfrac{t}{2}\right] F_i \cdot \ln \tfrac{2\exp\left[(u-w)^2 t/2\right]}{F_i} du & (3.8)\\
&\leq \tfrac{t}{w2^t} \cdot \int_{y=0}^{w} (\tfrac{w+y}{w})^t \cdot \exp\left[-y^2 \tfrac{t}{2}\right] F_i \cdot \ln \tfrac{2\exp\left[y^2 t/2\right]}{F_i} \cdot dy & (3.9)\\
&\leq \tfrac{t}{w2^t} \cdot \int_{y=0}^{w} \exp\left[\tfrac{y}{w}t - y^2\tfrac{t}{2}\right] F_i \cdot \ln \tfrac{2\exp\left[y^2 t/2\right]}{F_i} \cdot dy & \\
&\leq \tfrac{t}{w2^t} \cdot \int_{y=0}^{\infty} \exp\left[-\tfrac{t}{2}(y-\tfrac{1}{w})^2 + \tfrac{t}{2w^2}\right] F_i \cdot (\tfrac{y^2 t}{2} + 1 + \ln 1/F_i) \cdot dy & \\
&\leq \tfrac{\sqrt{t}}{w2^t} e^{t/2w^2} F_i \int_{z=0}^{\infty} \exp\left[\frac{-(z-\frac{\sqrt{t}}{w})^2}{2}\right] \left(\tfrac{z^2}{2} + 1 + \ln \tfrac{1}{F_i}\right) dz & (3.10)
\end{aligned}$$

where Eqn. (3.7) uses Eqn. (3.6), Eqn. (3.8) uses Fact 3.4.8 and Fact 3.4.9, and we make substitution $y = u - w$ in Eqn. (3.9) and substitution $z = \sqrt{t}y$ in Eqn. (3.10).

Using Fact 3.4.10 for $y_0 = \sqrt{t}/w \geq 1$, we can further bound $\mathbb{E}_{\tilde{h}}[f_i \ln 1/f_i]$ as follows

$$\begin{aligned}
\mathbb{E}_{\tilde{h}}[f_i \ln 1/f_i] - L_i &\leq \frac{\sqrt{t}}{w} 2^{-t} \exp\left[\frac{t}{2w^2}\right] F_i \cdot \left(O\left((\tfrac{\sqrt{t}}{w})^2\right) + O(1) + O(\ln 1/F_i)\right) \\
&\leq O\left(\left(\frac{\sqrt{t}}{w}\right)^3 \exp\left[\frac{t}{2w^2}\right]\right) \cdot 2^{-t} F_i \ln 2/F_i. & (3.11)
\end{aligned}$$

Also, using Fact 3.4.9 and the fact that $\mathbb{E}_{B_1,\ldots B_{i-1},PF_i}[f_i] \leq F_i$, the value of L_i is

$$\begin{aligned}
L_i &= \int_{v=0}^{w} \frac{\mathrm{Sur}^t(v)}{\mathrm{Vol}^t(2w)} \cdot \mathbb{E}_{B_1,\ldots B_{i-1},PF_i}[f_i] \ln \frac{1}{\mathbb{E}_{B_1,\ldots B_{i-1},PF_i}[f_i]} \cdot dv \\
&\leq \frac{\mathrm{Vol}^t(w)}{\mathrm{Vol}^t(2w)} \cdot F_i \ln 2/F_i \\
&\leq 2^{-t} F_i \ln 2/F_i. & (3.12)
\end{aligned}$$

It remains to plug-in Eqn. (3.11) and Eqn. (3.12) into Eqn. (3.6), and we obtain Proposition 3.4.7. □

To compute the entire value $I'(\tilde{h})$, we sum the contributions $\mathbb{E}_{\tilde{h}}[f_i \ln 1/f_i]$ over all $i \geq 1$ using Proposition 3.4.7:

$$\mathbb{E}_{\tilde{h}}\left[I'(\tilde{h})\right] \leq O\left(\left(\tfrac{\sqrt{t}}{w}\right)^3 \exp\left[\tfrac{t}{2w^2}\right]\right) \cdot 2^{-t} \textstyle\sum_{i\geq 1} F_i \ln 2/F_i.$$

It remains to prove that $2^{-t}\sum_{i\geq 1} F_i \ln 1/F_i \leq O(1)$, where $F_i = \exp\left[-(i-1)2^{-t}\right]$:

$$\begin{aligned} 2^{-t}\sum_{i\geq 1} F_i \ln 2/F_i &= 2^{-t}\sum_{i\geq 1} \exp\left[-(i-1)2^{-t}\right]\cdot\left(1+(i-1)2^{-t}\right) \\ &\leq 2^{-t}\cdot\sum_{j\geq 0}\exp\left[-j+1\right]\cdot(2^t+2^t j) \\ &\leq O(1), \end{aligned}$$

where $j = \lceil (i-1)2^{-t} \rceil$.

We can finally conclude that the entropy contribution $I'(\tilde{h})$ of the balls with centers inside $B(0^t, 2w)$ is at most $O(1)$. Thus, the total entropy $I(\tilde{h})$ is at most $O(1)+\exp\left[-\Omega(t)\right] \leq O(1)$. This finishes the proof of Lemma 3.4.5.

3.5 Bibliographic Notes

By now, several LSH families have been discovered. We briefly survey them in the section below. After that, we also describe some further work related to the NN problem.

3.5.1 LSH Library

For each LSH family, we present the procedure of choosing a random function from the respective LSH family, as well as its locality-sensitive properties.

Hamming distance. For binary vectors from $\{0,1\}^d$, Indyk and Motwani [IM98] propose LSH function $h_i(p) = p_i$ where $i \in \{1, \ldots d\}$ is a randomly chosen index (the sample LSH family from Section 3.1). They prove that the exponent ρ is $1/c$ in this case.

It can be seen that the above family applies directly to M-ary vectors (that is, with coordinates in $\{1 \ldots M\}$) under the Hamming distance. Moreover, a simple reduction enables to extend this family of functions to M-ary vectors under the ℓ_1 distance [LLR94]. Consider any point p from $\{1 \ldots M\}^d$. The reduction proceeds by computing a binary string Unary(p) obtained by replacing each coordinate p_i by a sequence of p_i ones followed by $M - p_i$ zeros. It is easy to see that for any two M-ary vectors p and q, the Hamming distance between Unary(p) and Unary(q) equals to the ℓ_1 distance between p and q. Unfortunately, this reduction is efficient only if M is relatively small.

ℓ_1 distance. A more direct LSH family for $\mathbb{R}^d$ under the ℓ_1 distance is described in [AI06a, And05]. Fix a real $w \gg R$, and impose a randomly shifted grid with cells of width w; each cell defines a bucket. More specifically, pick random reals $s_1, s_2, \ldots s_d \in [0, w)$, and define $h_{s_1,\ldots s_d}(x) = (\lfloor (x_1 - s_1)/w \rfloor, \ldots, \lfloor (x_d - s_d)/w \rfloor)$. The resulting exponent is equal to $\rho = 1/c + O(R/w)$.

ℓ_p distance. For the Euclidean space, [DIIM04] propose the following LSH family. Pick a random projection of $\mathbb{R}^d$ onto a 1-dimensional line, and chop the line into segments of length w, shifted by a random value $b \in [0, w)$. Formally, $h_{r,b}(x) = (\lfloor (r \cdot x + b)/w \rfloor$, where the projection vector $r \in \mathbb{R}^d$ is constructed by picking each coordinate of r from the Gaussian distribution. The exponent ρ drops strictly below $1/c$ for some (carefully chosen) finite value of w. This is the family used in the E^{2}LSH package [AI05].

A generalization of this approach to ℓ_p norms for any $p \in [0, 2)$ is possible as well; this is done by picking the vector r from a so-called *p-stable distribution*. Details can be found in [DIIM04].

Jaccard. To measure similarity between two sets $A, B \subset U$ (containing, say, words from two documents), the authors of [Bro97, BGMZ97] utilize the *Jaccard coefficient*. The Jaccard coefficient is defined as $s(A, B) = \frac{|A \cap B|}{|A \cup B|}$. Unlike the Hamming distance, Jaccard coefficient is a *similarity measure*: higher values of Jaccard coefficient indicate higher *similarity* of the sets. One can obtain the corresponding distance measure by taking $d(A, B) = 1 - s(A, B)$. For this measure, [Bro97, BGMZ97] propose the following LSH family, called *min-hash*. Pick a random permutation π on the ground universe U. Then, define $h_\pi(A) = \min\{\pi(a) \mid a \in A\}$. It is not hard to prove that the probability of collision $\Pr_\pi[h_\pi(A) = h_\pi(B)] = s(A, B)$. See [BCFM00] for further theoretical developments related to such hash functions.

Arccos. For vectors $p, q \in \mathbb{R}^d$, consider the distance measure that is the angle between the two vectors, $\theta(p, q) = \arccos\left(\frac{p \cdot q}{\|p\| \cdot \|q\|}\right)$. For this distance measure, Charikar (inspired by [GW95]) defines the following LSH family [Cha02]. Pick a random unit-length vector $u \in \mathbb{R}^d$, and define $h_u(p) = \mathrm{sign}(u \cdot p)$. The hash function can also be viewed as partitioning the space into two half-spaces by a randomly chosen hyperplane. Here, the probability of collision is $\Pr_u[h_u(p) = h_u(q)] = 1 - \theta(p, q)/\pi$.

ℓ_2 distance on a sphere. Terasawa and Tanaka [TT07] propose an LSH algorithm specifically designed for points that are on a unit hypersphere in the Euclidean space. The idea is to consider a regular polytope, orthoplex for example, inscribed into the hypersphere and rotated at random. The hash function then maps a point on the hypersphere into the closest polytope vertex lying on the hypersphere. Thus, the buckets of the hash function are the Voronoi cells of the polytope vertices lying on the hypersphere. [TT07] obtain exponent ρ that improves over [DIIM04] and the Leech lattice approach described in Section 3.3.2 (for general dimension $d > 24$).

3.5.2 Other related work

We also give a brief overview of prior work in the spirit of the algorithms mentioned in this chapter. Some of the papers considered a closely related problem of finding all "close" pairs of points in a data set. For simplicity, we translate them into the near neighbor framework, since they can be solved by performing essentially n separate near neighbor queries.

Hamming distance. Several papers investigated multi-index hashing-based algorithms for retrieving similar pairs of vectors with respect to the Hamming distance. Typically, the hash functions were projecting the vectors on some subset of the coordinates $\{1 \ldots d\}$, as in the example from earlier section. In some papers [PRR95, GPY94] the authors considered the probabilistic model where the data points are chosen uniformly at random, and the query point is a "random" point "close" to one of the points in the data set. A different approach [KWZ95] is to assume that the data set is arbitrary, but almost all points are far from the query point. Finally, the paper [CR93] proposed an algorithm which did not make any assumption on the input. The analysis of the algorithm was akin to the analysis sketched at the end of Section 3.1.1: the parameters k and L were chosen to achieve desired level of sensitivity and accuracy.

We also note that, recently, [Dub08] achieved an interesting bound for a probabilistic model of the Hamming space. That algorithm (for the "closest pair" problem) achieves an exponent of $\rho = \frac{1}{2c-1}$. We note that the algorithm is somewhat similar to the "ball partitioning" method we utilize in Section 3.2.1, although some details are different. Dubiner's exponent ρ matches the lower bound of [MNP06] for the Hamming distance at $c \to \infty$.

Set intersection measure. To measure the similarity between two sets A and B, the authors of [Bro97, BGMZ97] considered the Jaccard coefficient $s(A, B)$, proposing a family of hash functions $h(A)$ such that $\Pr[h(A) = h(B)] = s(A, B)$ (presented in detail in Section 3.5.1). Their main motivation was to construct short similarity-preserving "sketches" of sets, obtained by mapping each set A to a sequence $\langle h_1(A), ..., h_k(A)\rangle$. In Section 5.3 of their paper they briefly mention an algorithm similar to Strategy 2 described at the end of the Section 3.1.1. One of the differences is that, in their approach, the functions h_i are sampled *without replacement*, which made it more difficult to handle small sets.

Chapter 4

External Applications of LSH

In the previous chapter, we have shown how the Locality-Sensitive Hashing (LSH) scheme leads to efficient NN algorithms under the Hamming and Euclidean spaces. We now give indication of a broader applicability of the LSH scheme, beyond the NN application. In particular, in this chapter we show how LSH functions help for the problem of approximating kernel spaces, which may also be seen as dimensionality reduction in (implicit) kernel spaces.

Below, we first introduce and discuss the problem of approximating the kernel spaces, and then show how LSH leads to some efficient solutions. The results from this chapter have previously appeared in [AI08a].

4.1 Approximating Kernel Spaces

Kernel functions are a fundamental tool for learning a non-linear classifier. For example, they form a key component of Support Vector Machines (SVM). A kernel function defines a scalar product in a high-dimensional Euclidean space. Alternatively, it can be viewed as a lifting of the data space $\mathcal{S}$ into a new feature space, called the *kernel space* $\mathcal{K} \subset L_2$. The lifting enables performing complex classification using only a simple linear separator.

However, the map ϕ lifting the original space $\mathcal{S}$ into the kernel space is usually not explicit and the dimensionality of the kernel space is very high (or even infinite). As a result, algorithms that use the mapping ϕ directly are very inefficient. The classical approach this problem (the *kernel trick*) is to design algorithms that rely only on the scalar product in $\mathcal{K}$, given by the kernel function $K(x,y) = \phi(x) \cdot \phi(y)$ for all $x, y \in \mathcal{S}$ (see [MMR$^+$01, SS02b]).

Here, we seek to attack the problem more directly, by constructing *explicit* and *efficient* maps of the data space into the kernel space of *low dimension*. Specifically, our goal is to construct a map $F : \mathcal{S} \to \mathbb{R}^k$, for some *small* value of k, such that, for any $x, y \in \mathcal{S}$, the scalar product $F(x) \cdot F(y)$ is (approximately) equal to $K(x,y)$. This approach, in various forms, has been proposed before, e.g., in [Blu06, AMS01, DM03, BBV06, RR07].

The approach has multiple benefits (see [RR07]). First, one can compute the large-margin separator directly, using direct algorithms that are potentially more efficient. Second, the classification itself can be done much more efficiently. Specifically, in a standard approach, an SVM outputs a classifier[1] $f(x) = \sum_{i=1}^{S} \alpha_i K(x, x_i)$, where $\{x_1, \ldots x_S\}$ are the support vectors. Evaluating $f(x)$ takes time that is linear in the number of support vectors, which in principle could be as large as the number of the data points. In contrast, using the explicit map F, one can compute the weights w of a linear separator explicitly by letting

[1] An example x is classified as positive iff $f(x) > 0$.

$w = \sum_{i=1}^{S} \alpha_i F(x_i)$. Then the classifier can be defined as $f(x) = F(x) \cdot w$. The latter classifier can be evaluated in only $O(k)$ time, which is independent of the number of the support vectors.

The *existence* of a map F into a low-dimensional kernel space for any kernel can be derived from the random dimension-reduction techniques, such as Johnson-Lindenstrauss lemma (see Fact 2.3.2 in Section 2.3). Namely, if we project the high-dimensional kernel space into a random low-dimensional subspace, then the scalar product between any pair of unit vectors is preserved up to an additive term of ϵ. Then the map F is defined as a composition of the high-dimensional map ϕ and the random projection. Arriaga–Vempala [AV06] further prove that the resulting F also approximately preserves the separation margin between the two classes. Unfortunately, the aforementioned existential construction is highly inefficient, since it uses the original high-dimensional mapping $\phi : \mathcal{S} \to \mathcal{K}$. Instead, we would like to construct a map F directly.

The problem of designing efficient dimensionality reduction techniques of kernel spaces has been previously investigated in the literature. Some of the first results were obtained for a simpler problem of designing the map F that works for a particular purpose (e.g, linear classification) and for a given dataset. This question can be seen as approximating the kernel (Gram) matrix $M_{ij} = K(x_i, x_j)$ of some data set $D = \{x_1, \ldots x_n\}$ (see, e.g., [Blu06, BBV06, AMS01, DM03]). For example, [BBV06] consider the question of constructing F after one draws a small number of samples from the dataset and has only *black-box access* to $K(x, y)$. Under this condition, they construct a low-dimensional map F that preserves *linear separability* of the kernelized dataset. However, the constructed F depends on the data distribution[2]. Furthermore, the constructed mapping preserves linear separability of the data, but it does not appear to approximate the kernel function itself. Our more strict condition guarantees usefulness of F for other applications of kernels, such as regression and clustering. Because of these reasons, [Blu06, BBV06] asks if it is possible to construct data-independent F for specific kernels.

More recently, Rahimi–Recht [RR07] provide the only currently known data-independent constructions. They give two constructions for maps F that approximate the kernel space. Their first construction works for the case when data live in the Euclidean space and the kernel is shift-invariant, i.e., $K(x, y) = K(\|x - y\|_2)$. For $\mathcal{S} = \mathbb{R}^d$, their function F maps the data points into a space of dimension $k = O(d \cdot \frac{\log 1/\epsilon}{\epsilon^2})$ and can be evaluated in a similar time. The construction proceeds by defining each feature as a sinusoid with a parameter drawn from a distribution defined by the Fourier transform of the kernel function. Their second construction is designed specifically for the Laplacian kernel $L(x, y) = e^{-\|x-y\|_1}$. The latter construction computes each feature in two steps. First, a randomly-shifted grid is imposed on the space $\mathbb{R}^d$. Then a point $x \in \mathbb{R}^d$ is encoded as the id of the grid cell containing x, represented in unary. Their experiments show that both methods compare favorably with standard methods for classification.

4.2 LSH for Approximating Kernel Spaces

We now show how the LSH technique yields a generic theoretical framework for approximating kernel spaces. In particular, we illustrate how some of the existing LSH families give efficient low-dimensional explicit maps F for corresponding spaces $\mathcal{S}$ and kernels K. Our

[2] In fact, as [BBV06] prove, this condition is necessary if we have only a black-box access to the kernel function.

approach generalizes the second approach of [RR07]. However, our framework expresses a more general underlying phenomenon. As a result, we easily obtain mappings F for other similarity or dissimilarity functions.

4.2.1 Kernel hash functions

We start by defining the notion of a family of *kernel hash functions.* Before giving a formal definition, we explain the intuition. Ideally, we would like to obtain a distribution over hash functions h such that $\Pr_h[h(x) = h(y)] = K(x, y)$ for all $x, y \in \mathcal{S}$. However, such a guarantee might be hard (or impossible) to obtain in some cases. Instead, we introduce a relaxed notion, which we call a family of *ϵ-approximate kernel hash functions.*

Definition 4.2.1. *For $\epsilon > 0$ and kernel function K, we define* a family of ϵ-approximate K-kernel hash functions (KHF) *as a set $\mathcal{H}$ of functions $h : \mathcal{S} \to U$ for some set U if, for all $x, y \in \mathcal{S}$, we have*

$$\left| \Pr_{h\in\mathcal{H}}[h(x) = h(y)] - K(x, y) \right| \leq \epsilon.$$

To illustrate the definition, we consider an example of such a family $\mathcal{H}$ for some specific K and $\epsilon = 0$. This family $\mathcal{H}$ is based on the original LSH scheme of [IM98]. Consider the hypercube $\mathcal{S} = \{0, 1\}^d$ with the kernel function $K_p(x, y) = \left(1 - \frac{H(x,y)}{d}\right)^p$, where $p \in \mathbb{N}$ is a fixed positive integer, and $H(\cdot, \cdot)$ is the Hamming distance. We choose a hash function $h \in \mathcal{H}$ by taking a random set of coordinates $i_1, \ldots i_p \in [d]$ (with replacement), and setting $h(x) = x_{i_1} \cdots x_{i_p}$. In words, h is a projection to a random set of p coordinates. It is immediate to check that $\mathcal{H}$ satisfies the above definition for $\epsilon = 0$.

4.2.2 Kernel maps from approximate kernel hash functions

We now prove how, given a family of ϵ-approximate kernel hash functions, we obtain the desired map F lifting data space into an (approximate) low-dimensional kernel space. Intuitively, we construct $F(x)$ by sampling many $h_i \in \mathcal{H}$, for some family $\mathcal{H}$ of approximate kernel hash functions, and then concatenating $h_i(x)$'s.

Lemma 4.2.2. *Let $\epsilon > 0$. Fix a space $\mathcal{S}$ that admits a family $\mathcal{H}$ of ϵ-approximate K-kernel hash functions, for a kernel function K. For any $\delta > 0$, there exists a randomized mapping $F : \mathcal{S} \to \mathbb{R}^k$, where $k = O(\frac{\log 1/\delta}{\epsilon^2})$, such that, for any $x, y \in \mathcal{S}$, we have $|F(x) \cdot F(y) - K(x, y)| < 2\epsilon$ with probability at least $1 - \delta$.*

The time to compute $F(x)$ is bounded by the time to evaluate functions from $\mathcal{H}$, times k.

We note that the image of the mapping F has a very simple form: it is a scaled hypercube $\left\{-\frac{1}{\sqrt{k}}, +\frac{1}{\sqrt{k}}\right\}^k$.

Proof. Draw k functions h from $\mathcal{H}$ and call them $h_1, \ldots h_k$. Consider the function

$$F(x) = \tfrac{1}{\sqrt{k}} \cdot \langle E(h_1(x)), E(h_2(x)), \ldots E(h_k(x))\rangle,$$

where $E : U \to \ell_2$ is an "encoding function", mapping the universe U into vectors of reals. For now we assume that E is such that $E(a) \cdot E(b) = 1$ when $a = b$ and $E(a) \cdot E(b) = 0$ when $a \neq b$; we will relax this assumption later in the proof. Let $\chi[A] \in \{0, 1\}$ be the indicator random variable for an event A, which is equal to 1 iff A is true. Then, we can see that $F(x) \cdot F(y) = \frac{\sum_{i=1}^{k} \chi[h_i(x)=h_i(y)]}{k}$. Furthermore, by Chernoff bound, we have

$$\left| F(x) \cdot F(y) - \Pr_h[h(x) = h(y)] \right| \le \epsilon/2$$

with probability at least $1 - \delta/3$. Finally, using the definition of ϵ-approximate K-kernel hash functions $\mathcal{H}$, we deduce that $|F(x) \cdot F(y) - K(x, y)| \le \epsilon + \epsilon/2$ with probability at least $1 - \delta/3$.

It remains to describe the encoding function E. A simple approach is to encode the universe U in a unary format, that is, map symbols $a \in U$ into a vectors of length U with exactly one coordinate equal to 1. However this is inefficient, since it multiplies the target dimension k by $|U|$. Instead, for each coordinate $i \in [k]$, we choose a random map $E_i : U \to \{-1, +1\}$, and take

$$F(x) = \frac{1}{\sqrt{k}} \cdot \langle E_1(h_1(x)), E_2(h_2(x)), \ldots E_k(h_k(x)) \rangle.$$

It is easy to see that even after this simplification, $|F(x) \cdot F(y) - K(x, y)| \le 2\epsilon$ with probability at least $1 - \delta$. Indeed, let c be the number of indexes i such that $h_i(x) = h_i(y)$. Then $F(x) \cdot F(y)$ is a sum of c ones and $k - c$ independent random variables chosen uniformly at random from $\{-1, +1\}$. We have already shown that $|c/k - K(x, y)| \le \epsilon + \epsilon/2$. By Chernoff bound, the sum of the other $k - c$ values is at most $\epsilon/2 \cdot k$ with probability at least $1 - \delta/3$. The conclusion follows from an application of the triangle inequality. □

4.2.3 Some families of kernel hash functions

Next we show how to obtain (approximate) kernel hash functions for various kernels from the existing LSH families of functions. In fact, we show that most of the LSH families from Section 3.5.1 and the new LSH family from Section 3.2.1 give approximate KHFs. By Lemma 4.2.2, we immediately obtain efficient maps F into low-dimensional kernel spaces, for the corresponding kernels.

We defer the proofs of the lemmas from below to the next section for clarity.

Laplacian kernel. Consider the d-dimensional Manhattan space $\mathcal{S} = \ell_1^d$ and the Laplacian kernel $L(x, y) = e^{-\|x-y\|_1/\sigma}$, for some $\sigma > 0$. We show that, for any $\epsilon > 0$, this space admits a family of ϵ-approximate L-kernel hash functions based on the LSH functions of [AI06a, And05]. The final resulting map is similar to the second construction of [RR07].

We show how to pick a hash function $h \in \mathcal{H}$. Fix parameters $p = 2/\epsilon$ and $t = \sigma \cdot p$. Then construct p random functions f_i, $i = 1 \ldots p$, by imposing a randomly shifted regular grid of side length t. Formally, we choose $s_1, \ldots s_d$ at random from $[0, t)$, and define

$$f_i(x_1, \ldots, x_d) \triangleq (\lfloor (x_1 - s_1)/t \rfloor, \ldots, \lfloor (x_d - s_d)/t \rfloor).$$

The kernel hash function h is simply a concatenation of f_i chosen as above: $h(x) = (f_1(x), f_2(x), \ldots f_p(x))$.

Lemma 4.2.3. *Let $\epsilon > 0$. Suppose h is a hash function chosen as above. Then, for any $x, y \in \ell_1^d$, we have $\left| \Pr_h[h(x) = h(y)] - L(x, y) \right| \le \epsilon$.*

We note that the same result holds for the Laplacian kernel in the Euclidean space (instead of ℓ_1), namely $L_2(x,y) = e^{-\|x-y\|_2/\sigma}$. To obtain the family, we use the exact same hash functions as above except that, for each f_i, we rotate its grid at random beforehand.[3]

Near-Gaussian kernel. Consider the d-dimensional Euclidean space $\mathcal{S} = \ell_2^d$ and the kernel $K_{\text{erfc}}(x,y) = \frac{\text{erfc}(\|x-y\|_2/\sigma)}{2-\text{erfc}(\|x-y\|_2/\sigma)}$, where $\text{erfc}(x) = \frac{2}{\sqrt{\pi}}\int_x^\infty e^{-t^2}dt$ is the *Gauss error function*. As we show in a moment, this function approximates well the Gaussian kernel $e^{-\|x-y\|_2^2/\sigma^2}$.

A KHF for K_{erfc} follows from the LSH family from Section 3.2.1, which we recall here for clarity. Set $t = O(\frac{1}{\epsilon^2}\log 1/\epsilon)$ and $w = \frac{1}{2\sqrt{2}}\sqrt{t}\sigma$. First pick a random projection from $\mathbb{R}^d$ to $\mathbb{R}^t$, denoted by the matrix A. Then, in the projected t-dimensional space, pick $U = 2^{O(t\log t)}$ grids of balls of radius w, where a grid $u \in [U]$ of balls is the (infinite) set of balls with centers at $4w \cdot \mathbb{Z}^d + s_u$ for a random translation $s_u \in [0, 4w)^d$. Finally, define $h(x)$ as the index of the ball with the smallest $u \in [U]$ that contains the point Ax, the projection of x.

Lemma 4.2.4. *Let $\epsilon > 0$. Suppose h is a hash function chosen as above. Then, for any $x, y \in \ell_2^d$, we have $\left|\Pr_h[h(x) = h(y)] - K_{\text{erfc}}(x,y)\right| \le \epsilon$. The function h can be evaluated in time $2^{\tilde{O}(1/\epsilon^2)}$.*

We note that this same family can be used for the Gaussian kernel $G(x,y) = e^{-\|x-y\|_2^2/\sigma^2}$, although we do not achieve an approximation for arbitrary value of $\epsilon > 0$. However, the following lemma proves that the above family is 0.16-approximate family of G-kernel hash functions.

Lemma 4.2.5. *Suppose h is a hash function chosen as above for fixed $t = O(1)$ and $w = O(1)$. Then, for any $x, y \in \ell_2^d$, we have $\left|\Pr_h[h(x) = h(y)] - G(x,y)\right| \le 0.16$. The function h can be evaluated in constant time.*

Jaccard kernel. Consider the space of sets over some universe W, namely $\mathcal{S} = \{A : A \subseteq W\}$, under the kernel $K_J(A,B) = \frac{|A\cap B|}{|A\cup B|}$.

Here, a KHF follows from the standard min-hash functions designed by [Bro97, BGMZ97]. A hash function is chosen as follows. Pick a random permutation π on the ground universe W. Then, define $h_\pi(A) = \min\{\pi(a) \mid a \in A\}$. The family is 0-approximate kernel hash function.

Geodesic kernel. Consider a hypersphere in d-dimensional space $\mathcal{S} = \mathbb{S}^{d-1}$ with the kernel $K_\theta(x,y) = 1 - \frac{\theta(x,y)}{\pi}$, where $\theta(x,y)$ is the angle between vectors x and y which is proportional to the geodesic distance from x to y on the hypersphere.

Here, a KHF follows from the "random hyperplane" hash function designed by Charikar [Cha02] (inspired by [GW95]). A hash function is chosen as follows. Pick a random unit-length vector $u \in \mathbb{R}^d$, and define $h_u(x) = \text{sign}(u \cdot x)$. The hash function can also be viewed as partitioning the space into two half-spaces by a randomly chosen hyperplane passing through the center. The resulting family is a family of 0-approximate K_θ-kernel hash functions.

[3]This can be seen as embedding the problem over Euclidean space to one on ℓ_1 [JS82] and then using the above LSH.

4.2.4 Proofs of the KHF properties

Proof of Lemma 4.2.3. The proof follows directly from Lemma 4.1.1 in [And05], which states that for any f_i, $i = 1 \dots p$,

$$1 - \|x-y\|_1/t \le \Pr_{f_i}[f_i(x) = f_i(y)] \le e^{-\|x-y\|_1/t}.$$

Since $\Pr_h[h(x) = h(y)] = \prod_{i=1}^p \Pr_{f_i}[f_i(x) = f_i(y)]$, the probability of collision under h is

$$(1 - \|x-y\|_1/t)^p \le \Pr_h[h(x) = h(y)] \le e^{-\|x-y\|_1 p/t}.$$

If we let $\Delta = \|x-y\|_1/\sigma$, then $\|x-y\|_1/t = \Delta/p$. We use the approximation $e^{-\xi}e^{-\frac{\xi^2}{1-\xi}} \le 1 - \xi \le e^{-\xi}$ for $\xi \in (0,1)$. Then, for $\Delta/p \le 1/2$, we obtain

$$\left|\Pr_h[h(x) = h(y)] - e^{-\|x-y\|_1/\sigma}\right| \le e^{-\Delta} - e^{-\Delta} \cdot e^{-p\frac{(\Delta/p)^2}{1-\Delta/p}} \le e^{-\Delta}\left(1 - \left(1 - p\frac{(\Delta/p)^2}{1-\Delta/p}\right)\right) \le \frac{2}{p}\max_{\Delta \ge 0}\frac{\Delta^2}{e^{\Delta}}.$$

Since $\max_{\Delta \ge 0} \Delta^2/e^{\Delta} \le 1$ and $p = 2/\epsilon$, the above quantity is upper-bounded by ϵ. For $\Delta > p/2 = 1/\epsilon$, the conclusion follows immediately since, in this case, $e^{-\|x-y\|_1/\sigma} < \epsilon$. □

Proof of Lemma 4.2.4. We use the analysis of this hash function from Section 3.2.2. First, Lemma 3.2.2 proves that the entire space $\mathbb{R}^t$ will indeed be covered by balls with probability at least $1 - 2^{-\Omega(t \log t)} \ge 1 - \epsilon/4$. Second, we argue that after the projection into $\mathbb{R}^t$ is performed, the incurred distortion of $\|x-y\|_2$ is negligible. Indeed, let $\Delta = \|x-y\|_2$. Johnson-Lindenstrauss lemma says that $\Delta' = \|Ax - Ay\|_2$ is within a multiplicative factor of $1+\epsilon/8$ of Δ (see Section 2.3). Then, $|K_{\text{erfc}}(x,y) - K_{\text{erfc}}(Ax, Ay)| = |K_{\text{erfc}}(\Delta) - K_{\text{erfc}}(\Delta')| = |K'_{\text{erfc}}(\xi)| \cdot |\Delta - \Delta'| \le \epsilon/3$ where $\xi \in [\Delta(1-\epsilon/8), \Delta(1+\epsilon/8)]$.

Finally, Eqn. (3.1) states that

$$\Pr[h(Ax) = h(Ay) \mid \|Ax - Ay\|_2 = \Delta'] = \frac{I(\Delta'/2, w)}{1 - I(\Delta'/2, w)},$$

where $I(\Delta'/2, w)$ is the probability that a random point chosen from a ball of radius w has its first coordinate at least as big as $\Delta'/2$. We approximate the distribution of the first coordinate of a random point from a ball as a Gaussian of variance $\frac{1}{t}w^2$, using an estimate from [DF87]. The probability that a random Gaussian of variance $\frac{1}{t}w^2$ is greater than $\Delta'/2$ is precisely $\frac{1}{2}\operatorname{erfc}(\Delta' \cdot \frac{\sqrt{t}}{2\sqrt{2}w})$. Thus, we obtain that $|I(\Delta'/2, w) - \frac{1}{2}\operatorname{erfc}(\Delta' \cdot \frac{\sqrt{t}}{2\sqrt{2}w})| \le 16/t \le \epsilon/20$ (see [DF87]). This further implies that $\left|\frac{I(\Delta'/2,w)}{1-I(\Delta'/2,w)} - K_{\text{erfc}}(\Delta')\right| \le \epsilon/4$.

In the end, we conclude that $|\Pr_h[h(x) = h(y)] - K_{\text{erfc}}(x,y)| \le \epsilon$. □

Proof of Lemma 4.2.5. We observe that $\max_{\Delta \in [0,\infty)} \left|e^{-\Delta^2} - \frac{\operatorname{erfc}(0.4\Delta)}{2 - \operatorname{erfc}(0.4\Delta)}\right| \le 0.158$. The lemma then follows by Lemma 4.2.4 for $\epsilon = 0.001$. □

Chapter 5

NN via Product Spaces

In this chapter, we present our new approach to NN, based on embeddings into somewhat unusual type of spaces, namely, *iterated product spaces*. One of the concrete consequence of this approach is a new NN algorithm for the Ulam distance, a variant of edit distance (see Section 2.1 for definitions and motivation). For strings of length d, we obtain an NN algorithm that achieves $\tilde{O}(\log\log d)$ approximation with $dn^{1+\epsilon}$ space and dn^{ϵ} query time, for any fixed $\epsilon > 0$. We note that this beats the best possible bounds that can be achieved via the classical approaches to Ulam distance. Specifically, embeddings into ℓ_1, ℓ_2, powers thereof, ℓ_∞, or constant-sized sketches all lead to $\tilde{\Omega}(\log d)$ approximation and/or inefficient NN algorithms, as is described in Chapters 1 and 7.

Recall that product spaces may be seen as combinations of the standard ℓ_1, ℓ_2, and ℓ_∞ norms. For example, ℓ_1-product of (l copies of) k-dimensional ℓ_∞ is a new norm, denoted as $\bigoplus_{\ell_1}^{l} \ell_\infty^k$. The points in this space are $l \times k$ matrices with the following norm: take ℓ_∞ norm of each row, and having obtained l positive reals, take their ℓ_1 norm to obtain the desired $\bigoplus_{\ell_1}^{l} \ell_\infty^k$ norm. (See details in Section 2.1.)

Our NN algorithm for the Ulam distance essentially follows from two steps, proved in this chapter:

- Ulam distance on strings of length d embeds into the iterated product space $\bigoplus_{(\ell_2)^2}^{O(d)} \bigoplus_{\ell_\infty}^{O(\log d)} \ell_1^{O(d)}$, with constant distortion;
- the $\bigoplus_{(\ell_2)^2}^{O(d)} \bigoplus_{\ell_\infty}^{O(\log d)} \ell_1^{O(d)}$ space admits an efficient NN with a (roughly) double-logarithmic approximation.

These two steps substantiate our more general message — that the iterated product spaces may give a better balance between *richness* and *tractability* as a host space of an embedding. Indeed, first, the constant distortion embedding of Ulam distance indicates that the (iterated) product spaces are richer than any of its separate components, ℓ_1, squared-ℓ_2, or low-dimensional ℓ_∞.

Second, we demonstrate that there are good solutions for the NN algorithm for the iterated product spaces. In fact, we will show something more general: it is possible to achieve NN algorithms for *any* iterated product space of any combination of ℓ_p norms (for a fixed number of iterations), with only $(\log\log n)^{O(1)}$ approximation.

Furthermore, using the new approach we also obtain an improved NN algorithm for the planar Earth-Mover Distance metric. This metric also suffers from the same bottleneck as the Ulam/edit distances in the context of classical approaches to it.

Finally, to give further evidence of the versatility of the product spaces, in the next chapter, we will show how product spaces play an important role in obtaining algorithms for other applications, such as approximating edit distance in near-linear time.

In the rest of the chapter, we describe the new NN algorithms for product spaces, as well as the embedding of Ulam distance into product spaces. We then combine the two steps to obtain the new NN algorithm for the Ulam distance. We also describe how to obtain a new NN algorithm for EMD. The results from this chapter have previously appeared in [AIK09].

5.1 New Algorithms for Iterated Product Spaces

We now design efficient algorithms for iterated product spaces. For example, to obtaining an NN algorithm for Ulam with $\tilde{O}(\log\log d)$ approximation, we develop an NN algorithm for the space $\bigoplus_{(\ell_p)^p}^{k} \bigoplus_{\ell_\infty}^{l} \ell_1^m$ for $p \in [1, 2]$.

Our main ingredient is a new NN scheme designed for an ℓ_p-product metric $\bigoplus_{\ell_p} \mathcal{M}$, for all $p \in [1, \infty)$. This latter scheme uses a technique, which we call *black-box* Locality Sensitive Hashing. As described in Chapter 3, LSH-type techniques have been used before for NN under simple metrics like ℓ_1 and ℓ_2, and are based on probabilistic partitions of the corresponding space. Naturally, for a metric like $\bigoplus_{\ell_1} \mathcal{M}$, we cannot hope to do a similar partitioning of the space since we do not have any information about the metric $\mathcal{M}$. However, we show the space can be partitioned in a black-box manner so as to effectively reduce NN for $\bigoplus_{\ell_1} \mathcal{M}$ to NN for the max-product $\bigoplus_{\ell_\infty} \mathcal{M}$, with a mild increase in parameters. For the latter max-product $\bigoplus_{\ell_\infty} \mathcal{M}$, we can use the algorithms of [Ind98, Ind02b]. We note that a related idea was also present in [Ind04]. However, the algorithm of [Ind04] had much larger (super-logarithmic) approximation factor, which makes it inapplicable to the scenarios we consider here.

We note that our approach also gives NN algorithms for ℓ_p spaces for $p \in (2, \infty)$, by setting $\mathcal{M} = \mathbb{R}$. No algorithms for such spaces were previously known. Before continuing with the main theorem statement, we define the following a natural "scaling" property of a metric space.

Definition 5.1.1. *Let $(\mathcal{M}, d_{\mathcal{M}})$ be a metric space. A map $\sigma : \mathcal{M} \to \mathcal{M}$ is called an α-*dilation*, for $\alpha > 0$, if for all $x, y \in \mathcal{M}$ we have $d_{\mathcal{M}}(\sigma(x), \sigma(y)) = \alpha \cdot d_{\mathcal{M}}(x, y)$. The metric is called* scalable *if for every $\alpha > 0$ it has an α-dilation σ_α. To simplify notation, we write $\alpha \cdot x$ for $\sigma_\alpha(x)$.*

We now show how to reduce an ℓ_p-product metric to an ℓ_∞-product metric, for any $p \in [1, \infty)$. Note that for $p = 1$, this corresponds to an ℓ_1-product $\bigoplus_{\ell_1} \mathcal{M}$.

Theorem 5.1.2 (NN for ℓ_p-product)**.** *Let $(\mathcal{M}, d_{\mathcal{M}})$ be a scalable metric space where computing distance take $\tau_{\mathcal{M}}$ time. Also let $k \geq 1$. Suppose there is an NN scheme for the max-product $\bigoplus_{\ell_\infty}^{k} \mathcal{M}$ with approximation c, query time $Q(n)$, and space $S(n)$. Then for every $p \in [1, \infty)$, and $\epsilon > 0$, there is an NN scheme for the ℓ_p-product $\bigoplus_{\ell_p}^{k} \mathcal{M}$ with approximation $\tilde{c} = \frac{2}{\epsilon} c$, query time $O(n^{\epsilon^p}) \cdot (Q(n) + k\tau_{\mathcal{M}})$, and space $O(n^{\epsilon^p}) \cdot S(n)$.*

We note that we can combine the above theorem with an existing algorithm for the ℓ_∞-product of a metric of [Ind02b, Ind98]. We describe this in more detail once we prove Theorem 5.1.2.

Proof of Theorem 5.1.2. On an intuitive level, we design a generalization of the Locality Sensitive Hashing (LSH), introduced in Chapter 3. As mentioned there, LSH has been used to design NN under simple metrics like ℓ_1 and ℓ_2. Remember that LSH is a (non-adaptive) hashing scheme that probabilistically partitions the entire space into buckets such that a pair of "close" points (distance $\leq R$) have higher probability of collision (i.e., falling into the same bucket) than a pair of "far" points (distance $> \tilde{c}R$). The LSH algorithm then builds several hash tables, each hashing all n data points according to a fresh random partition. Upon receiving a query q, the algorithm computes the hash of q and linearly scans the data points that fall into the same bucket and reports those that are indeed close to q.

Ideally, we would like to be able to similarly partition the space $\bigoplus_{\ell_p} \mathcal{M}$, however we cannot do this since we have no control over $\mathcal{M}$. (Moreover, when $p > 2$, we do not even know of any LSH functions for the simple ℓ_p norm, much less for an ℓ_p-product.) Nonetheless, we manage to do so in a black-box manner, as will be seen later, replacing a hash table structure by a nearest neighbor data structure for $\bigoplus_{\ell_\infty} \mathcal{M}$. Our algorithm may be viewed as a (distant) generalization of the LSH scheme for ℓ_1 described in the LSH Library in Section 3.5.1. We now describe our algorithm in detail.

Let $\varepsilon = \epsilon/2 \in (0, \frac{1}{2})$. Note that $\tilde{c}$ becomes $\tilde{c} = c/\varepsilon$ in the new notation.

Fix a threshold radius $R > 0$. Let $f_p(z) = \frac{pz^{p-1}}{w} e^{-z^p/w}$ be the derivative of $e^{-z^p/w}$, where $w = (R/\varepsilon)^p / \ln 2n^{1/(1-\varepsilon^p)}$. Note that $f_p(z)$ defines a probability distribution on $z \in [0, \infty)$ since $\int_0^\infty f_p(z)dz = 1$ and $f_p(z) \geq 0$.

Preprocessing stage. For $L = 10n^{\varepsilon^p/(1-\varepsilon^p)} \leq 10n^{(2\varepsilon)^p}$, construct L different max-product data structures (these correspond to the L hash tables of an LSH scheme). For each $i \in [L]$, construct one max-product data structure M_i, as follows. Pick reals $s_1^i, s_2^i, \ldots s_k^i$ each from the distribution f_p. From the dataset D, construct the dataset $\tilde{D}_i$ containing all $\tilde{x}$ such that $\tilde{x}$ is obtained from $x \in D$ by scaling each coordinate $j \in [k]$ in the product by $1/s_j^i$. In other words, if $x_j \in \mathcal{M}$ is the j^{th} coordinate of $x \in D \subseteq \mathcal{M}^k$, then

$$\tilde{D}_i = \left\{ \tilde{x} = \left(x_1/s_1^i, x_2/s_2^i, \ldots x_k/s_k^i \right) \mid x \in D \right\}.$$

Finally we let M_i be a near-neighbor data structure for $\bigoplus_{\ell_\infty}^k \mathcal{M}$ metric with the threshold set to $R' = 1$, constructed on the dataset $\tilde{D}_i$.

Query algorithm. Given a query point $q \in \mathcal{M}^k$, iteratively go over M_i for all $i \in [L]$. For each M_i, compute $\tilde{q}^i = (q_1/s_1^i, q_2/s_2^i, \ldots, q_k/s_k^i)$ and query the point $\tilde{q}^i$ in M_i. For each returned point x, compute the actual distance from q to x and report the point x if $d_{p,\mathcal{M}}(x, q) \leq \tilde{c}R$, where $\tilde{c} = c/\varepsilon$. Once such a point is reported, stop.

Correctness and running time. The analysis is somewhat reminiscent of the one used for the standard LSH scheme. Fix one NN data structure M_i. We prove that, for each M_i, none of the "far" (under original metric) points x may be returned by M_i, with probability at least 1/2. Also, each R-near neighbor (under the original metric) is a near neighbor in some M_i (under the M_i's metric, the max-product), with constant probability. Both of these facts allows us to conclude the correctness and query time bound.

We start by computing the probability P_2 that a "far" point x, with $d_{p,\mathcal{M}}(x, q) > \tilde{c}R$, becomes a c-near neighbor in the M_i data structure, for the query $\tilde{q}^i$. In other words, the probability P_2 is equal to the probability that the point $\tilde{x}^i = \left(x_1/s_1^i, x_2/s_2^i, \ldots x_k/s_k^i\right)$ is within $d_{\infty,\mathcal{M}}$ distance c of $\tilde{q}^i$ (in the terms of an LSH scheme, this event is a "collision"). Let $\delta_j = d_{\mathcal{M}}(x_j, q_j)$ for $j \in [k]$. Note that $\sum_j \delta_j^p = (d_{p,\mathcal{M}}(x,q))^p > (\tilde{c}R)^p$ by the definition of x.

Now, for each coordinate $j \in [k]$, we have $d_{\mathcal{M}}(\tilde{x}_j^i, \tilde{q}_j^i) \le c$ if and only if $\delta_j / s_j^i \le c$, or, equivalently, $s_j^i \ge \delta_j / c$. Thus, the probability that $d_{\infty,\mathcal{M}}(\tilde{x}^i, \tilde{q}^i) \le c$ is precisely equal to

$$\Pr\left[d_{\infty,\mathcal{M}}(\tilde{x}^i, \tilde{q}^i) \le c\right] = \prod_{j=1}^{k} \int_{z=\delta_j/c}^{\infty} f_p(z)dz = \prod_{j=1}^{k} e^{-(\delta_j/c)^p/w} < e^{-(\tilde{c}R/c)^p/w} = \tfrac{1}{2}n^{-1/(1-\varepsilon^p)}.$$

Thus we get that $P_2 \le n^{-1/(1-\varepsilon^p)}/2$. Similarly, one can compute P_1, defined as the probability that $\|\tilde{x}^i - \tilde{q}^i\| \le 1$ given that $d_{p,\mathcal{M}}(x,q) \le R$:

$$P_1 \ge \Pr\left[d_{\infty,\mathcal{M}}(\tilde{x}^i, \tilde{q}^i) \le 1\right] = \prod_{j=1}^{k} \int_{z=\delta_j}^{\infty} f_p(z)dz \ge e^{-R^p/w} = 2^{-\varepsilon^p} n^{-\varepsilon^p/(1-\varepsilon^p)}.$$

Now, let F be the set of dataset points that are "far", i.e., are at distance more than $\tilde{c}R$ from q. Let C be the set of R-near neighbors of q.

Then, for each data structure M_i, in expectation, only at most $n^{-1/(1-\varepsilon^p)}/2 \cdot n = n^{-\varepsilon^p/(1-\varepsilon^p)}/2$ points from F become a possible answer for M_i. By a Markov's bound, with probability at least $1 - n^{-\epsilon^p/(1-\varepsilon^p)}/2$, none of the points from F are a possible answer for M_i. Furthermore, if there exists an R-near neighbor $x \in C$, then this point x remains a 1-NN under M_i's distance (and is thus a valid answer for M_i) with probability at least $P_1 \ge 2^{-\varepsilon^p} n^{-\varepsilon^p/(1-\varepsilon^p)}$. Thus, an M_i returns some $\tilde{c}R$-near neighbor of q with probability at least $2^{-\varepsilon^p} n^{-\varepsilon^p/(1-\varepsilon^p)} - n^{-\epsilon^p/(1-\varepsilon^p)}/2 \ge 0.2 \cdot n^{-\varepsilon^p/(1-\varepsilon^p)}$. In total, over all L data structures M_i, the probability of reporting some $\tilde{c}R$-near neighbor becomes at least $1 - (1 - (0.2 \cdot n^{-\varepsilon^p/(1-\varepsilon^p)}))^L \ge 1 - e^{-0.2 \cdot 10} \ge 0.8$.

Since we can implement each M_i with query time $Q(n)$ and space $S(n)$, the final data structure has query time $L \cdot (Q(n) + O(k\tau_{\mathcal{M}})) = O(n^{(2\varepsilon)^p}) \cdot (Q(n) + k\tau_{\mathcal{M}})$, and similarly space is $O(n^{(2\varepsilon)^p}) \cdot S(n)$. Replacing $\varepsilon = \epsilon/2$, we obtain the desired bounds. □

We now prove our most general NN result: that any iterated product space of ℓ_p's admits an efficient NN with only $(\log\log n)^{O(1)}$ approximation. We first define iterated product spaces more formally: a t-iterated space is the following space, for some $q \in [1, \infty)$, $p_1, \ldots p_t \in [1, \infty]$, and $k_1, \ldots k_t \in \mathbb{N}$:

$$\mathcal{S} = \bigoplus\nolimits_{(\ell_{p_1})^q}^{k_1} \bigoplus\nolimits_{\ell_{p_2}}^{k_2} \cdots \bigoplus\nolimits_{\ell_{p_{t-1}}}^{k_{t-1}} \ell_{p_t}^{k_t}.$$

For example, 1-iterated spaces include $\ell_1, \ell_2, \ell_\infty$ and their powers. We note that, without loss of generality, we need not consider higher powers of norms beyond the first product.

We prove the following theorem.

Theorem 5.1.3. *For any $\epsilon > 0$, there exists an NN algorithm for the space $\mathcal{S}$ with the following parameters:*

- $O((\log\log n)^t \cdot (\epsilon/t)^{-t-1/p_1-1/p_2-\ldots-1/p_t})^q$ *approximation,*
- $O(n^\epsilon \cdot k_1 k_2 \ldots k_t)$ *query time, and*
- $O(n^{1+2\epsilon} \cdot (k_1 k_2 \ldots k_t)^2)$ *space and preprocessing.*

The theorem follows from combining, recursively, the above theorem with the following NN algorithm for an ℓ_∞-product of [Ind02b, Ind98].

Theorem 5.1.4 ([Ind02b, Theorem 1], [Ind98])**.** *Consider metric $(\mathcal{M}, d_{\mathcal{M}})$ with an NN algorithm that achieves approximation c, query time $Q(n)$, and space $S(n)$.*[1] *Then, for every $\epsilon > 0$, there exists NN under $\bigoplus_{\ell_\infty}^k \mathcal{M}$ with:*

- *$O(\epsilon^{-1} \log\log n)$ approximation,*
- *$O((Q(n) + k\tau_{\mathcal{M}}) \log n)$ query time, where $\tau_{\mathcal{M}}$ is the time to compute distance in $\mathcal{M}$, and*
- *$S(n) \cdot O(kn^{1+\epsilon})$ space/preprocessing, and, if $S(n) \le \lambda \cdot n^{1+\rho}$ for some λ and ρ, then the space/preprocessing becomes $O(\lambda k^2 n^{1+\epsilon+\rho})$.*

For the ℓ_∞ case (when $\mathcal{M} = \mathbb{R}$), the approximation becomes $O(\epsilon^{-1} \log\log k)$, with $O(k \log n)$ query time and $O(kn^{1+2\epsilon})$ space/preprocessing.

Proof of Theorem 5.1.3. We design an NN for the q^{th} root of the space $\mathcal{S}$, namely:

$$\mathcal{S}' = \bigoplus_{\ell_{p_1}}^{k_1} \bigoplus_{\ell_{p_2}}^{k_2} \cdots \bigoplus_{\ell_{p_{t-1}}}^{k_{t-1}} \ell_{p_t}^{k_t} = \bigoplus_{\ell_{p_1}}^{k_1} \bigoplus_{\ell_{p_2}}^{k_2} \cdots \bigoplus_{\ell_{p_{t-1}}}^{k_{t-1}} \bigoplus_{\ell_{p_t}}^{k_t} \mathbb{R},$$

which is a metric. We now apply Theorems 5.1.2 and 5.1.4 in order, for t times, with the following parameters ϵ: $\epsilon^{1/p_1}, \epsilon, \epsilon^{1/p_2}, \epsilon, \ldots$. Also, we note that $\mathbb{R}$ admits a trivial efficient NN for 2-approximation, with $O(1)$ query time, and $O(n)$ space: just chop up the line into segments of length R, store one dataset point per segment, and, for a query q, check its segment as well as the left/right segments. We obtain the following parameters for $\mathcal{S}'$:

- $O(\log\log n)^t \cdot \epsilon^{-1/p_1-1-1/p_2-1-\ldots-1/p_t-1}$ approximation,
- $O(n^{t\epsilon} \cdot k_1 k_2 \ldots k_t)$ query time, and
- $O(n^{1+2t\epsilon} \cdot (k_1 k_2 \ldots k_t)^2)$ space and preprocessing.

The theorem follows from scaling ϵ and raising the approximation to power q to get the approximation under the distance of the space $\mathcal{S}$. $\square$

5.2 Embedding Ulam Distance into Product Spaces

We now show that the Ulam distance embeds well into product spaces. Combined with the NN algorithms for product spaces from the previous section, we will obtain greatly improved NN algorithms for the Ulam distance; we will describe these algorithms in the next section.

We present two embedding of the Ulam distance into product spaces. The first one is a constant distortion embedding of Ulam distance into $\bigoplus_{(\ell_p)^p}^d \bigoplus_{\ell_\infty}^{O(\log d)} \ell_1^{2d}$ for every fixed $p \in (1, 2]$.

Theorem 5.2.1. *For every $d \in \mathbb{N}$ and $p = 1 + \epsilon$ where $\epsilon > 0$, there exists an embedding $\varphi : \mathrm{Ulam}_d \mapsto \bigoplus_{(\ell_p)^p}^d \bigoplus_{\ell_\infty}^{O(\log d)} \ell_1^{2d}$ such that for all $x, y \in \mathrm{Ulam}_d$:*

$$\mathrm{ed}(x, y) \le d_{p^p,\infty,1}(\varphi(x), \varphi(y)) \le O(\tfrac{1}{\epsilon}) \cdot \mathrm{ed}(x, y).$$

[1] Strictly speaking, we need to impose a technical condition on the NN for $\mathcal{M}$, but it is satisfied in all our scenarios; see [Ind02b, Section 2] for details.

When φ is viewed as an embedding into $\ell_1^{O(d^2 \log d)}$, it has distortion $O(\log^2 d)$. The image $\varphi(x)$ of an input string x can be computed in time $O(d^2 \log d)$.

This embedding alone already yields an approximation of $(\log\log n)^{O(1)}$ for the NN under the Ulam distance, via the general Theorem 5.1.3. To further improve the approximation to depend only on d (as opposed to n), we design a second embedding, which is derived from the first one. The second embedding has the advantage of a somewhat simpler host space, $\bigoplus_{(\ell_p)^p} \ell_\infty$, however, it handles only one scale of distances and has high dimension (for our intended use).

Lemma 5.2.2. *For every $1 \le R, \alpha \le d$ there is a randomized map $\hat{\varphi} : \mathrm{Ulam}_d \to \bigoplus_{(\ell_p)^p}^{d^3} \ell_\infty^m$ with $m = d^{O(\alpha)}$, such that for every $x, y \in \mathrm{Ulam}_d$, and $p = 1 + \epsilon$ for $\epsilon > 0$, with probability at least $1 - e^{-\Omega(d^2)}$ we have:*

- *if $\mathrm{ed}(x, y) \ge R$ then $d_{p^p,\infty}(\hat{\varphi}(x), \hat{\varphi}(y)) \ge \Omega(R)$, and*
- *if $\mathrm{ed}(x, y) \le R/\alpha$ then $d_{p^p,\infty}(\hat{\varphi}(x), \hat{\varphi}(y)) \le O(\epsilon^{-1} \cdot R/\alpha)$.*

We note that the main tools behind our embeddings also lead to improved algorithms for the problem of *sublinear distance estimation* for Ulam distance and a variant of *smoothed* (standard) edit distance — see details in the bibliographic notes at the end of the chapter.

5.2.1 First embedding of the Ulam distance

We now give our embeddings of the Ulam distance into $\bigoplus_{(\ell_p)^p}^{d} \bigoplus_{\ell_\infty}^{O(\log d)} \ell_1^{2d}$, thus proving Theorem 5.2.1.

Our new embedding of Ulam metric is based on a new estimate of Ulam's distance. It is inspired by previous work on testing and estimating the *distance to monotonicity/sortedness* [EKK+00, ACCL07, GJKK07], but unlike the previous estimates which are asymmetric in the two strings, our new estimate is entirely symmetric in the two strings. Our estimate uses primitives such as "count", "there exists", and "majority". Although the last two primitives do not look very "geometric", our estimate will be such that it can be transformed into a distance function, in fact a norm, defined via iterated product spaces. The resulting embedding turns out to be very simple to compute, mapping a string into a carefully chosen collection of incidence vectors of its substrings.

Before continuing to the proofs, we introduce some additional notation. As mentioned in Section 2.1, Ulam_d denotes the Ulam metric over strings of length d over alphabet Σ; we assume, for simplicity of presentation that $\Sigma = [d]$. For $P, Q \in \mathrm{Ulam}_d$, we let $\underline{\mathrm{ed}}(P, Q)$ denote the minimum number of deletions from P to obtain a subsequence of Q. Note that $\underline{\mathrm{ed}}(Q, P) = \underline{\mathrm{ed}}(P, Q)$ and $\underline{\mathrm{ed}}(P, Q) \le \mathrm{ed}(P, Q) \le 2\,\underline{\mathrm{ed}}(P, Q)$. For $x \in \Sigma^d$, we use the notation x_i or $x[i]$ to refer to the i^{th} position in x.

We start by presenting the construction of the embedding φ.

Construction of φ. We use the following notation. For $P \in \mathrm{Ulam}_d$, we assume by convention that in positions $j = 0, -1, \ldots, -d+1$ we have $P[j] = j$ and set the *extended* alphabet to be $\bar{\Sigma} = \{-d+1, \ldots, d\}$. For $a \in [d]$ and $k \in [d]$, let P_{ak} be a set containing the k symbols that appear in the k positions immediately before symbol a in P, i.e. $P_{ak} = \{P[P^{-1}[a] - k], \ldots, P[P^{-1}[a] - 1]\}$.

We proceed in three steps. First, for a symbol $a \in [d]$ and integer $k \in [d]$, we define $\varphi_{ak} : \mathrm{Ulam}_d \mapsto \ell_1^{2d}$ by setting $\varphi_{ak}(P)$ to be the 0/1 incidence vector of P_{ak} scaled by

$1/2k$. Thus, $\varphi_{ak}(P) \in \{0, \frac{1}{2k}\}^{\bar{\Sigma}}$ and has exactly k nonzero entries. Distances in this host space are computed using the ℓ_1-norm, namely $\|\varphi_{ak}(P) - \varphi_{ak}(Q)\|_1$. Second, for every $a \in \Sigma$, define $\varphi_a : \mathrm{Ulam}_d \mapsto \bigoplus_{\ell_\infty}^{O(\log d)} \ell_1^{2d}$ to be the direct sum $\varphi_a(P) = \oplus_{k \in K} \varphi_{ak}(P)$, where $K = \{\lceil (1+\gamma)^i \rceil : i = 0, 1, \ldots, \lceil \log_{1+\gamma} d \rceil\}$ ranges over all powers of $1+\gamma$ in $[d]$ where we set $\gamma = 1/4$.[2] Distances in this product space are computed using an ℓ_∞-norm, namely $\mathsf{d}_{\infty,1}(\varphi_a(P), \varphi_a(Q)) = \max_{k \in K} \|\varphi_{ak}(P) - \varphi_{ak}(Q)\|_1$. Third, define $\varphi : \mathrm{Ulam}_d \mapsto \bigoplus_{(\ell_p)^p}^{d} \bigoplus_{\ell_\infty}^{O(\log d)} \bigoplus_{\ell_1}^{2d}$ by the direct sum $\varphi(P) = \oplus_{a \in [d]} \varphi_a(P)$. Distances in this host space are computed using $(\ell_p)^p$, i.e. $\mathsf{d}_{p^p,\infty,1}(\varphi(P), \varphi(Q)) = \sum_{a \in [d]} \left(\mathsf{d}_{\infty,1}(\varphi_a(P), \varphi_a(Q))\right)^p$.

An estimate of the Ulam distance. The following lemma is key to the proof of Theorem 5.2.1 and provides an estimate on the Ulam distance between two permutations. It is inspired by, and technically builds upon, [EKK$^+$00, ACCL07, GJKK07], which gave estimates to the distance from P to a fixed permutation, say the identity (hence called distance to monotonicity). Relabeling of the symbols can clearly be used to apply these previous estimates to two arbitrary permutations P and Q; however, it requires an explicit description of Q^{-1}, which is inefficient or just impossible, in our intended applications.[3] Thus, the main advantage of our estimate is that it is the first one which is *efficient* for two *arbitrary* permutations. In the sequel, we use $A \triangle B$ to denote the symmetric difference between two sets A, B.

Lemma 5.2.3. *Fix $P, Q \in \mathrm{Ulam}_d$, and let $0 < \delta \le 1/2$. Let T_δ be the set containing all symbols $a \in \Sigma$ for which there exists $k \in [d]$ such that the symmetric difference $|P_{ak} \triangle Q_{ak}| > 2\delta k$. Then*

$$\frac{1}{2}\, \underline{\mathrm{ed}}(P, Q) \le |T_\delta| \le \frac{4}{\delta} \cdot \underline{\mathrm{ed}}(P, Q). \tag{5.1}$$

In the case of two permutations P and Q, there is a crucial (albeit technical) difference between our estimate and the previous ones, including [ACCL07, GJKK07]. The core of all such estimates is a count of the number of symbols that satisfy a particular "easy to compute" property with respect to the rest of the strings. In our case, for a given symbol a and an integer k, the property is based on the value of $|P_{ak} \triangle Q_{ak}|$ (the symmetric difference between the k symbols appearing immediately before a in P and similarly in Q); and it is well-known that symmetric difference can be expressed as the ℓ_1 difference between the respective incidence vectors. In contrast, previous estimates are based on the count of how many of the k symbols appearing immediately before a in P (i.e. the set P_{ak}), appear in Q *after* a. Such a set-intersection formulation does not lend itself to embeddings. In this sense, our estimate is symmetric with respect to P and Q, while previous ones are not. Nevertheless, our proof relies on the technical analysis of [ACCL07, GJKK07], but in a rather nontrivial way. In particular, we restore symmetry between the two permutations by applying the known bounds twice, once from P towards Q and once from Q towards P. The full proof of Lemma 5.2.3 follows.

Proof of Lemma 5.2.3. Fix $P, Q \in \mathrm{Ulam}_d$ and $0 < \delta \le 1/2$. We say that two distinct symbols $a, b \in \Sigma$ are inverted in P vs. in Q if these symbols do not appear in the same

[2] To simplify the exposition, we shall ignore rounding issues and the fact that the largest value in K should be capped by d.

[3] We seek embeddings that are oblivious, i.e., the image of P has to be determined independently of Q. In the NN algorithms, the data string P are preprocessed without knowing the query Q. Sublinear algorithms cannot afford to compute Q^{-1} explicitly, as it would take linear time.

order in P and in Q, i.e. if $(P^{-1}[a]-P^{-1}[b])(Q^{-1}[a]-Q^{-1}[b])<0$. We say that a pair of indexes i,j in P is inverted if the respective symbols $P[i],P[j]$ are inverted. Define a set R_δ^P containing all indexes $i\in[d]$ for which there is $j<i$ such that for more than δ-fraction of indexes $j'\in[j,i-1]$ the pair of indexes i,j' is inverted in P. We know from [GJKK07, Lemma 3.1] that

$$\underline{\mathrm{ed}}(P,Q)\le 2|R_{1/2}^P|. \tag{5.2}$$

(It is assumed therein that Q is the identity permutation; the bound above may seem more general but it follows immediately by relabeling symbols.) We claim that $R_{1/2}^P\subseteq R_\delta^P\subseteq T_\delta$. Indeed, whenever $a\in R_{1/2}^P$, there is $j<i$ such that more than $1/2\ge\delta$ of the indexes $j'\in[j,i-1]$ are inverted with respect to i in P, and in particular $P[j']\in P_{a,i-j}\setminus Q_{a,i-j}$. Since $|P_{a,i-j}|=|Q_{a,i-j}|=i-j$, it follows that $|P_{a,i-j}\triangle Q_{a,i-j}|=2|P_{a,i-j}\setminus Q_{a,i-j}|>2\delta(i-j)$, and thus $a\in T_\delta$, proving the claim. Using the claim and (5.2), we have $\underline{\mathrm{ed}}(P,Q)\le 2|T_\delta|$, which proves the first inequality in (5.1).

We proceed to proving the second inequality in (5.1). Fix an optimal alignment between P and Q, namely a subset $D\subseteq\Sigma$, $|D|=\underline{\mathrm{ed}}(P,Q)$ such that deleting the symbols of D from P and from Q yields identical strings. Let $D^P=\{i\in[d]:\ P[i]\in D\}$ denote the indexes of D in P, and define D^Q similarly for Q. Define a set S_δ^P containing all indexes $i\in[d]$ for which there is $j<i$ such that more than δ-fraction of indexes $j'\in[j,i-1]$ belong to D^P. Let S_δ^Q be defined similarly for Q. We then know from [GJKK07, Lemma 3.2],[4] that for all $0<\delta'\le 1/2$,

$$|R_{\delta'}^P\setminus D^P|\le|S_{\delta'}^P|\le(1-\delta')/\delta'\cdot|D^P|,$$

and, in fact, that $R_{\delta'}^P\setminus D^P\subseteq S_{\delta'}^P$. Therefore we deduce that

$$|R_{\delta'}^P\cup S_{\delta'}^P|\le|D^P\cup S_{\delta'}^P|\le\tfrac{1}{\delta'}\cdot|D^P|, \tag{5.3}$$

and similarly for Q.

We next show that

$$|T_\delta|\le|R_{\delta/2}^P\cup S_{\delta/2}^P|+|R_{\delta/2}^Q\cup S_{\delta/2}^Q|. \tag{5.4}$$

Indeed, consider $a\in T_\delta$ and let $k\in[d]$ be its witness, namely $|P_{ak}\triangle Q_{ak}|>2\delta k$. The case where $P^{-1}[a]\in R_{\delta/2}^P\cup S_{\delta/2}^P$ can be paid for using the term $|R_{\delta/2}^P\cup S_{\delta/2}^P|$. The case where $Q^{-1}[a]\in R_{\delta/2}^Q\cup S_{\delta/2}^Q$ can be paid for using the term $|R_{\delta/2}^Q\cup S_{\delta/2}^Q|$. We now claim that these are the only two possible cases, i.e. not being in either of the two cases implies a contradiction. Indeed, if $a\in T_\delta$ and $P^{-1}[a]\notin R_{\delta/2}^P\cup S_{\delta/2}^P$, then there must be at least one symbol $b'\in\bar\Sigma$ such that **(a)** $b'\in P_{ak}\setminus Q_{ak}$; **(b)** b' is not inverted wrt to a; and **(c)** b' is not in D. Using **(a)** and **(b)** we have that **(d)** b' appears in Q more than k positions before a (i.e. its index in Q is smaller than $Q^{-1}[a]-k$). Since also $Q^{-1}[a]\notin R_{\delta/2}^Q\cup S_{\delta/2}^Q$, we similarly obtain a symbol $b''\in\bar\Sigma$ such that **(a')** $b''\in Q_{ak}$; **(c')** b'' is not in D; and **(d')** b'' appears in P more than k positions before a. We obtain from **(a)** and **(d')** that b' appears after b'' in P, and from **(a')** and **(d)** that b'' appears after b' in Q. Thus, the symbols b',b'' are inverted, and at least one of them must belong to D, contradicting **(c)** and **(c')**. This proves the claim and (5.4).

Finally, using (5.3), (5.4), and the fact that $|D^P|=|D^Q|=\underline{\mathrm{ed}}(P,Q)$, we conclude $|T_\delta|\le\frac{2}{\delta}|D^P|+\frac{2}{\delta}|D^Q|=\frac{4}{\delta}\,\underline{\mathrm{ed}}(P,Q)$, which proves the second inequality in (5.1), and completes the proof of Lemma 5.2.3. ∎

[4] A similar upper bound, up to constant factors, is implied by results of [ACCL07, Lemma 2.3].

We can now complete the proof of Theorem 5.2.1 using Lemma 5.2.3. We need to bound the distortion of φ when viewed as an embedding into $\bigoplus_{(\ell_2)^2} \bigoplus_{\ell_\infty} \ell_1$. In a nutshell, the distortion of the embedding roughly corresponds to $\sum_{\delta=2^{-j}} \delta^p |T_\delta| / \operatorname{ed}(P,Q) \leq O(\sum_{\delta=2^{-j}} \delta^{p-1}) \leq O(1/(p-1))$, where the squared term comes from the outer $(\ell_p)^p$-product, and would not work if instead we were to use ℓ_1 as the outer product.

Proof of Theorem 5.2.1. Fix two distinct permutations $P, Q \in \mathrm{Ulam}_d$. By definition, for all $a \in \Sigma$ and $k \in [d]$ we have $\|\varphi_{ak}(P) - \varphi_{ak}(Q)\|_1 = \frac{1}{2k} |P_{ak} \triangle Q_{ak}|$.

We first bound $\mathsf{d}_{p^p,\infty,1}(\varphi(P), \varphi(Q))$ from below. By Lemma 5.2.3 (and its notation) for $\delta = 1/2$, we know that $|T_\delta| \geq \frac{1}{2} \underline{\mathrm{ed}}(P,Q)$. Now fix $a \in T_\delta$. Then there exists $k \in [d]$ such that $|P_{ak} \triangle Q_{ak}| > 2\delta k$, and rounding this k upwards to the next power of $1+\gamma$, we obtain $k' \in K$ such that $\|\varphi_{ak'}(P) - \varphi_{ak'}(Q)\|_1 = \frac{1}{2k'} |P_{ak'} \triangle Q_{ak'}| \geq \frac{1}{2k'}(2\delta k - 2\gamma k) = \frac{\delta - \gamma}{1+\gamma} = \frac{1}{5}$. (We remind that the rounding issues we neglected would lead to slightly worse constants.) Thus, for each $a \in T_\delta$ we have $\mathsf{d}_{\infty,1}(\varphi_a(P), \varphi_a(Q)) \geq 1/5$, and thus

$$\mathsf{d}_{p^p,\infty,1}(\varphi(P), \varphi(Q)) \geq \sum_{a \in T_\delta} (\tfrac{1}{5})^p \geq \tfrac{\underline{\mathrm{ed}}(P,Q)/2}{25} = \tfrac{\underline{\mathrm{ed}}(P,Q)}{50}.$$

To bound $\mathsf{d}_{p^p,\infty,1}(\varphi(P), \varphi(Q))$ from above, we relax the range $k \in K$ into $k \in [d]$, and break the contribution arising from different $a \in \Sigma$ into buckets of the form $[2^{-j}, 2^{-j+1}]$. Remember that $p = 1 + \epsilon$ for some small $\epsilon > 0$.

$$\begin{aligned}
\mathsf{d}_{p^p,\infty,1}(\varphi(P), \varphi(Q)) &= \sum_{a \in \Sigma} \Big(\mathsf{d}_{\infty,1}(\varphi_a(P), \varphi_a(Q)) \Big)^p \\
&\leq \sum_{a \in \Sigma} \max_{k \in [d]} \|\varphi_{ak}(P) - \varphi_{ak}(Q)\|_1^{1+\epsilon} \\
&= \sum_{a \in \Sigma} \max_{k \in [d]} \left[\tfrac{1}{2k} |P_{ak} \triangle Q_{ak}| \right]^{1+\epsilon} \\
&\leq 1 + \sum_{j=1}^{\log d} (2^{-j+1})^{1+\epsilon} \cdot |T_{2^{-j}}|.
\end{aligned}$$

By Lemma 5.2.3, we have $|T_{2^{-j}}| \leq 2^{j+2} \cdot \underline{\mathrm{ed}}(P,Q)$, and therefore

$$\mathsf{d}_{p^p,\infty,1}(\varphi(P), \varphi(Q)) \leq 1 + \sum_{j=1}^{\log d} 2^{-\epsilon j + 4} \cdot \underline{\mathrm{ed}}(P,Q) \leq O(\tfrac{1}{\epsilon}) \cdot \underline{\mathrm{ed}}(P,Q).$$

The second part of the theorem results from a similar computation on $\|\varphi(P) - \varphi(Q)\|_1$.

$$\begin{aligned}
\|\varphi(P) - \varphi(Q)\|_1 &= \sum_{a \in \Sigma} \|\varphi_a(P) - \varphi_a(Q)\|_1 \\
&\leq O(\log d) \cdot \sum_{a \in \Sigma} \max_{k \in K} \|\varphi_{ak}(P) - \varphi_{ak}(Q)\|_1 \\
&\leq O(\log d) \cdot \left(1 + \sum_{j=1}^{\log d} 2^{-j+1} \cdot |T_{2^{-j}}| \right) \\
&\leq O(\log^2 d) \cdot \underline{\mathrm{ed}}(P,Q).
\end{aligned}$$

$\square$

5.2.2 Second embedding of the Ulam distance

We now give our second embeddings of the Ulam distance that proves Lemma 5.2.2.

Proof of Lemma 5.2.2. We start with the embedding φ from Theorem 5.2.1, and construct from it $\hat{\varphi}$ using, along the way, the Ulam characterization from Lemma 5.2.3. There are two steps in the construction. The main (second) step is to do a randomized dimensionality reduction in the inner ℓ_1^d to ℓ_1 of dimension only $l = O(\alpha \log d)$, using [KOR00]-type sketches. Then we can embed ℓ_1^l into ℓ_∞ of dimension $m = 2^l$ using the classical isometric embedding (see, e.g., [Ind01a]). This step alone, however, will increase the distance by accumulating a lot of "spurious noise", generated by the approximation errors of the dimensionality reduction in ℓ_1 for the outer coordinates $a \in T_\delta$ with small δ's. To overcome this roadblock, in the first step, we need to do a randomized reorganization of the coordinates so that we have a good control over the noise coming from T_δ's in comparison to the legitimate coordinates contributing to the distance.

Let φ be the embedding from Theorem 5.2.1.

First step. We construct $\varphi' : \mathrm{Ulam}_d \to \bigoplus_{(\ell_p)^p}^{d^3} \bigoplus_{\ell_\infty}^{O(d \log d)} \ell_1^d$ from φ with the property that, for any $x, y \in \mathrm{Ulam}_d$, with probability $1 - e^{-\Omega(d^2)}$, we have that

$$\Omega(\min\{d^3, d^3 \cdot \tfrac{\mathrm{ed}(x,y)}{R}\}) \le d_{p^p,\infty,1}(\varphi'(x), \varphi'(y)) \le O((p-1)^{-1} \cdot \min\{d^3, d^3 \cdot \tfrac{\mathrm{ed}(x,y)}{R}\}). \quad (5.5)$$

Specifically, for each coordinate $i \in [d^3]$, randomly pick a set $S_i \subseteq [d]$ by including each index $a \in [d]$ with probability $1/R$. Then construct $\varphi'_i = \oplus_{a \in S_i} \varphi_a$. Note that we can view φ'_i as $\varphi'_i(x) \in \bigoplus_{\ell_\infty}^{|S_i|} \bigoplus_{\ell_\infty}^{O(\log d)} \ell_1^d \subseteq \bigoplus_{\ell_\infty}^{O(d \log d)} \ell_1^d$. We use the notation $\varphi'_{i,a} = \varphi_a$.

We show that φ' satisfies Eqn.(5.5). Fix some particular x, y at distance $\mathrm{ed}(x, y) = Q \le R$. Define T'_δ, for $\delta > 0$, as the number of coordinates i such that $d_{\infty,1}(\varphi'_i(x), \varphi'_i(y)) \ge \delta$. We will prove that $T'_\delta \le O(\frac{1}{\delta} \cdot \frac{Q \cdot d^3}{R})$, similarly to what Lemma 5.2.3 claims for T_δ's. Indeed, the probability that $d_{\infty,1}(\varphi'_i(x), \varphi'_i(y)) \ge \delta$ is bounded by the probability that $S_i \cap T_\delta \ne \emptyset$. Thus,

$$\Pr[d_{\infty,1}(\varphi'_i(x), \varphi'_i(y)) \ge \delta] \le 1 - (1 - 1/R)^{|T_\delta|} \le 1 - (1 - 1/R)^{O(Q/\delta)} \le O(\tfrac{Q}{\delta R}).$$

By Chernoff bound over all d^3 coordinates i, we conclude that $T'_\delta \le O(\frac{Q}{\delta R} \cdot d^3)$ with probability at least $1 - e^{-\Omega(d^2)}$. Analogously, it is easy to see that $T'_{1/2} = \min\{1, Q/R\} \cdot \Omega(d^3)$.

Given the bounds on T'_δ's, we deduce Eqn. (5.5) as we did in the Theorem 5.2.1 for φ.

Second step. We now view φ' obtained from the last step as $\varphi'(x) \in \bigoplus_{(\ell_p)^p}^{d^3} \bigoplus_{\ell_\infty}^{d} \bigoplus_{\ell_\infty}^{O(\log d)} \ell_1^d$. Thus, for each outer product coordinate $i \in [d^3]$, second-product coordinate $a \in S_i$, and third-product coordinate $k \in [K]$, the vector $\varphi_{i,a,k}$ is a d-dimensional Hamming vector scaled by $1/2k$. Consider the mapping $\psi_{i,a,k} : \{0,1\}^d \to \{0,1\}^l$, where each coordinate $j \in [l]$ of $\psi_{i,a,k}$ is defined as follows. For each character $b \in [d]$, pick an indicator $r_{j,b} \in \{0,1\}$ that is equal to one with probability $\delta_k = \frac{1}{4k}$. Then, for a vector $v \in \{0,1\}^d$, we define $\psi_{i,a,k}(v) = \left((\sum_{b \in [d]} v_b r_{j,b}) \mod 2 \right)_{j \in [l]}$.

Finally we can define the embedding $\hat{\varphi} : \mathrm{Ulam}_d \to \bigoplus_{(\ell_p)^p}^{d^3} \bigoplus_{\ell_\infty}^{O(d \log d)} \ell_\infty^{2^l}$. Let $\eta : \ell_1^l \to \ell_\infty^{2^l}$ be the isometric map from ℓ_1^l to $\ell_\infty^{2^l}$ (see, e.g., [Ind01a]). We define $\hat{\varphi}$ as

$$\hat{\varphi}(x) = \left(\tfrac{R}{d^3}\right)^{1/p} \cdot \oplus_{i\in[d^3]} \oplus_{a\in S_i} \oplus_{k\in[K]} \eta(\tfrac{1}{l} \cdot \psi_{i,a,k}(2k \cdot \varphi_{a,k}(x))).$$

In words, we start with the embedding φ', and, for each fixed i, a, k, we consider the ℓ_1 vector $\varphi'_{i,a,k} = \varphi_{a,k}$ (corresponding to innermost ℓ_1), and do a dimensionality reduction on it using $\psi_{i,a,k}$. $\psi_{i,a,k}$ performs l scalar products of $2k\varphi'_{i,a,k} \in \{0,1\}^d$ with random vectors r_j. We then normalize the resulting vector by multiplying it with $1/l$. The obtained vector in $\{0,1\}^l$ is then mapped into $\ell_\infty^{2^l}$ using η. Finally, we scale the entire vector with $(R/d^3)^{1/p}$ so that, say, distance that were d^3 before become R (we need to take the p^{th}-root of R/d^3 because the outer product is $(\ell_p)^p$ that raises the multipliers to power p).

Next, we prove that $\hat{\varphi}$ satisfies lemma statement. Fix some x, y at distance Q. Define T''_δ to be the set of all ($(\ell_p)^p$-product) coordinates i such that $\|\hat{\varphi}_i(x) - \hat{\varphi}_i(y)\|_\infty \geq \delta$.

First, suppose $Q = R/\alpha$. As before, we will prove that $|T''_\delta| \leq O(\frac{1}{\delta}) \cdot \frac{Q \cdot d^3}{R}$. For $\delta \leq 1/\alpha$, this is immediate since $|T''_\delta| \leq d^3$. Now consider δ such that $1/\alpha \leq \delta \leq 1/4$, and fix a coordinate $i \in [d^3] \setminus T'_\delta$. Then, for any $a \in S_i, k \in [K]$, the expected value of $\|\psi_{i,a,k}(x) - \psi_{i,a,k}(y)\|_1$ is $l \cdot \frac{1}{2}(1 - (1 - \frac{1}{4k})^{2k\cdot\delta}) \leq \frac{\delta l}{2}$. By Chernoff bound, $\|\psi_{i,a,k}(x) - \psi_{i,a,k}(y)\|_1 < \delta l$ with probability at least $1 - e^{-\Omega(\delta l)} \geq 1 - d^{-3}$. By union bound, with probability at least $1 - d^{-1}$, for all $a \in S_i$ and $k \in [K]$, we have $\frac{1}{l}\|\psi_{i,a,k}(x) - \psi_{i,a,k}(y)\|_1 < \delta$. Finally, using a Chernoff bound over all $i \in [d^3] \setminus T'_\delta$, we can conclude that, for any $\delta' \geq \delta$, we have that $|T''_{\delta'}| \leq |T'_{\delta'}| + O(d^3 \cdot 1/d)$, with probability at least $1 - e^{-\Omega(d)}$. Thus $|T''_\delta| \leq O(\frac{1}{\delta}) \cdot \frac{Q \cdot d^3}{R}$. As before, we obtain $\mathsf{d}_{p^p,\infty}(\hat{\varphi}(x), \hat{\varphi}(y)) \leq O((p-1)^{-1}Q)$. The bound extends immediately to the case when $Q \leq R/\alpha$.

Now let's consider the case when $Q = R$. We need to argue that $T''_c = \Omega(d^3)$ for some constant $c = \Omega(1)$. Then $\mathsf{d}_{p^p,\infty}(\hat{\varphi}(x), \hat{\varphi}(y)) \geq c^2 \cdot |T''_c| \cdot \frac{R}{d^3} = \Omega(R)$. Indeed, consider any $i \in T'_{1/2}$; note that $T'_{1/2} \geq \Omega(d^3)$. Then, as in the previous paragraph, $\frac{1}{l}\|\psi_{i,a,k}(x) - \psi_{i,a,k}(y)\|_1 \geq cl$ with probability at least $1 - 1/d^3$. By Chernoff bound over all $i \in T'_{1/2}$, we conclude that $T''_c \geq \Omega(d^3) - O(d^3/d)$ with probability at least $1 - e^{-\Omega(d^2)}$. The argument extends to $Q \geq R$ standardly. □

5.3 NN for the Ulam and EMD Distances

We now devise NN algorithms for the Ulam and EMD distances.

NN for Ulam distance. Our NN algorithm for Ulam distance follows by combining the above two embeddings into product spaces with the NN algorithms for the product spaces developed in Section 5.1. We will prove the following theorem.

Theorem 5.3.1. *For every constant* $\epsilon > 0$ *there is a randomized NN algorithm under* Ulam_d *that achieves* $O(\epsilon^{-4} \log\log d \cdot \log\log\log d)$ *approximation, with* $d^{O(1)}n^\epsilon$ *query time and* $d^{O(1)}n^{1+\epsilon}$ *space.*

Proof. First we show how to obtain the NN algorithms for the host spaces of the two Ulam embeddings, namely $\bigoplus_{(\ell_p)^p} \bigoplus_{\ell_\infty} \ell_1$ and $\bigoplus_{(\ell_p)^p} \ell_\infty$. While we could use the general Theorem 5.1.3, we will design NN algorithms for these spaces a bit more carefully, in order to get sharper bounds.

Claim 5.3.2. *For every $k, l, m \in \mathbb{N}$, $p \in [1,2]$, and $\epsilon > 0$, the space $\bigoplus_{(\ell_p)^p}^k \bigoplus_{\ell_\infty}^l \ell_1^m$ admits an NN algorithm achieving $O(\epsilon^{-1-2p}(\log\log n)^p)$ approximation, $(klm)^{O(1)} n^\epsilon$ query time, and $(klm)^{O(1)} n^{1+\epsilon}$ space.*

Proof. We design an NN algorithm for the p^{th} root of the desired space, namely, $\bigoplus_{\ell_p}^k \bigoplus_{\ell_\infty}^l \ell_1^m$ (which is a metric). Note that if obtain some approximation α for the latter metric, then this is also an α^p approximation for the desired space.

We start by noting that ℓ_1^m admits NN with approximation $1/\epsilon$, query time $O(mn^\epsilon)$ and space $O(mn^{1+\epsilon})$ via the corresponding LSH-based NN algorithm from the LSH Library in Section 3.5.1.

Next, we apply Theorem 5.1.2 to reduce the NN under $\bigoplus_{\ell_p}^k \bigoplus_{\ell_\infty}^l \ell_1^m$ to NN under $\bigoplus_{\ell_\infty}^k \bigoplus_{\ell_\infty}^l \ell_1^m = \bigoplus_{\ell_\infty}^{kl} \ell_1^m$. For the latter space $\bigoplus_{\ell_\infty}^{kl} \ell_1^m$ we use the max-product algorithm of [Ind02b], namely Theorem 5.1.4.

Thus, we obtain the following parameters for NN under $\bigoplus_{\ell_p}^k \bigoplus_{\ell_\infty}^l \ell_1^m$: $\alpha = O(\epsilon^{-1/p}\epsilon^{-2}\log\log n)$ approximation, $O((klm)^{O(1)} n^{2\epsilon}\log n)$ query time, and $O((klm)^{O(1)} n^{1+3\epsilon})$ space/preprocessing. The claim follows. □

Also, for $\bigoplus_{(\ell_p)^p} \ell_\infty$ we obtain the following, better NN algorithm.

Claim 5.3.3. *For every $k, l \in \mathbb{N}$, $p \in [1,2]$, and $\epsilon > 0$, the space $\bigoplus_{(\ell_p)^p}^k \ell_\infty^l$ admits an NN scheme achieving approximation $O(\epsilon^{-1-p}(\log\log kl)^p)$, query time $(kl)^{O(1)} n^\epsilon$, and space $(kl)^{O(1)} n^{1+\epsilon}$.*

Proof. The proof is very similar to the above one. We focus on the metric $\bigoplus_{\ell_p}^k \ell_\infty^l$. Using Theorem 5.1.2, we reduce NN under this latter metric to NN under $\bigoplus_{\ell_\infty}^k \ell_\infty^l = \ell_\infty^{kl}$. For the metric ℓ_∞^{kl}, we use the original ℓ_∞ algorithm of [Ind98] (see Theorem 5.1.4), which gives an approximation of $O(\log\log kl)$. □

It just remains to show how to combine the two embeddings for the Ulam distance with the above two data structures. Let $p = 1 + \frac{1}{\log\log\log d}$, in which case $(\log\log d)^p = O(\log\log d)$, and both embeddings have a distortion of $(p-1)^{-1} = O(\log\log\log d)$.

We employ one of two different data structures, depending on the parameters d and n. If $n \le d^{\log d}$, we apply Theorem 5.2.1 to embed Ulam into $\bigoplus_{(\ell_p)^p}^{O(d)} \bigoplus_{\ell_\infty}^{O(\log d)} \ell_1^{O(d)}$, and use the NN from Claim 5.3.2, achieving approximation $O(\epsilon^{-1-2p}(\log\log n)^p \cdot \log\log\log d) = O(\epsilon^{-4}\log\log d \cdot \log\log\log d)$ for the NN under Ulam distance.

If $n > d^{\log d}$, then we use Lemma 5.2.2 with $\alpha = O((p-1)^{-1}\log\log d) < O(\sqrt{\log d})$, embedding Ulam into $\bigoplus_{(\ell_p)^p}^{d^3} \ell_\infty^m$ for $m = d^{\sqrt{\log d}}$. Then we can apply Claim 5.3.3, which achieves $O(\log\log d)$ approximation for the $\bigoplus_{(\ell_p)^p}^{d^3} \ell_\infty^m$ space. The overall approximation for the NN under Ulam becomes $O((1-p)^{-1} \cdot \epsilon^{-1-p}\log\log(d^3 m)) = O(\epsilon^{-4}\log\log d \cdot \log\log\log d)$, with query time $O(d^{\sqrt{\log d}} n^\epsilon) \le O(n^{\epsilon + o(1)})$.

This completes the construction of the NN under Ulam distance and the proof of Theorem 5.3.1. □

NN for EMD. For EMD over $[d]^2$, our techniques, together with an embedding from [Ind07], yield NN with $O(\frac{\alpha}{\epsilon}\log\log n)$ approximation, $d^{O(1)} n^\epsilon$ query time and $n^{1+\epsilon} \cdot 2^{d^{1/\alpha}}$ space for any desired $\alpha = \alpha(d,n) > 1$ and $\epsilon > 0$. This improves upon the $O(\log d)$ approximation of [Cha02, IT03, NS07], albeit at the cost of a much higher space. Our approximation also beats the $\Omega(\sqrt{\log d})$ non-embeddability lower bound into ℓ_1 as long as $n \ll \exp\left[e^{\sqrt{\log d}}\right]$ [NS07].

Our general approach for designing NN under EMD follows the same principle as the above one for Ulam: we embed EMD into a product space, and then show an NN algorithm for the product space. For EMD, we use an existing embedding into a sum-product of smaller EMDs that was already given in [Ind07] to obtain a near-linear time distance estimation algorithm.

We note that, very recently, a somewhat better bound for the NN under EMD has been obtained in [ADIW09]. The result of [ADIW09] also uses the same embedding into product space, however, instead of designing an NN algorithm for it directly, that algorithm sketches the product space and then uses the sketch-based full-indexing NN algorithm.

Let EMD_d be the EMD metric over the two-dimensional grid $[d]^2$.

Theorem 5.3.4 ([Ind07, Theorem 3.1])**.** *For every $d \geq 1$, $\alpha > 0$, there exists a probabilistic embedding $\eta : \mathrm{EMD}_d \to \bigoplus_{\ell_1}^{m} \mathrm{EMD}_{d^{1/\alpha}}$, where $m = d^{O(1)}$, with approximation $O(\alpha)$. Namely, for every $\pi_A, \pi_B \in \mathrm{EMD}_d$, we have $d_{1,\mathrm{EMD}}(\eta(\pi_A), \eta(\pi_B)) \geq \mathrm{EMD}(\pi_A, \pi_B)$, and $\mathbb{E}_\eta\left[d_{1,\mathrm{EMD}}(\eta(\pi_A), \eta(\pi_B))\right] \leq O(\alpha) \cdot \mathrm{EMD}(\pi_A, \pi_B)$.*

Using this embedding, and Theorem 5.1.2, we obtain the following corollary.

Corollary 5.3.5. *For every constant $\epsilon > 0$, and every $d, n, \alpha \geq 1$, there exists NN under EMD_d that has $O(\alpha \cdot \epsilon^{-2} \log\log n)$ approximation, with $d^{O(1)} n^{\epsilon}$ query time, and $n^{1+\epsilon} \cdot 2^{d^{1/\alpha}}$ space.*

Proof. The result follows by applying Theorem 5.1.2 to the above embedding η, together with the algorithm from Theorem 5.1.4 for max-products, which is $\bigoplus_{\ell_\infty} \mathrm{EMD}_{d^{1/\alpha}}$ in our case. Note that, for EMD of size $[d^{1/\alpha}]^2$, we can easily construct a 2-approximation NN algorithm with $2^{O(d^{2/\alpha} \log d)}$ space and $O(d^{2/\alpha})$ query time by first discretizing the distributions to obtain multi-sets and then building a complete index on EMD over multi-sets. Rescaling α yields the desired result. □

5.4 Bibliographic Notes

Sub-linear time Ulam distance estimation. Our new Ulam "characterization" also leads to new efficient algorithms for the problem of *sub-linear time distance estimation*. In this problem, we are given two strings, x and y we the goal is to compute (approximately) the Ulam distance between x and y. We strive for a best possible algorithm, ideally one running in *sub-linear time*, i.e., one that does not even need to read the strings entirely. While this is generally not possible (imagine x and y differ in one position), it becomes possible whenever the distance $R = \mathrm{ed}(x,y)$ is relatively large. This case is of interest, for example, as a (heuristic) algorithm used for filtering in sequence alignment tools in order to weed out sure non-matches (see, e.g., [CNBYM01]).

An application of the characterization from Lemma 5.2.3 gives an $\tilde{O}(d/\sqrt{R})$ time algorithm for estimating Ulam distance between two strings of length d up to a $O(1)$ factor approximation. With more effort, in a later result [AN10], we further improve the time bound to $\tilde{O}(d/R + \sqrt{d})$, which is an *optimal* bound up to poly-logarithmic factors. The sublinear time algorithms for Ulam distance have found applications for the problem of sublinear distance estimation for a certain smoothed model of the (standard) edit distance; see details in [AK08b].

Product spaces. Product spaces were studied in the context of developing NN algorithms for other hard metrics in [Ind02b, Ind04]. An algorithm for NN under the max-product $\bigoplus_{\ell_\infty}^k \mathcal{M}$ is designed in [Ind02b], achieving $O(c \log\log n)$ approximation using polynomial space and sublinear query time, under the assumption that $\mathcal{M}$ itself has an NN scheme achieving c-approximation with polynomial space and sublinear query time (see Theorem 5.1.4). Although [Ind04] gave two algorithms for NN under the sum-product $\bigoplus_{\ell_1}^k \mathcal{M}$, they are much less satisfying, since one requires very large storage and the other obtains a rather large approximation. Our NN algorithm significantly improves the NN for sum-products from [Ind04], achieving performance comparable to that of max-products.

We note that the algorithms from [Ind04] were used to design algorithms for the NN under the edit distance. However, they did not provide any embedding of the edit distance into a simpler space, and thus do not fall under our approach of identifying richer host spaces. There has also been work on *streaming* product metrics such as $\bigoplus_{\ell_p} \ell_q$ (see, [CM05b, JW09]). Furthermore, product spaces, even iterated ones, are examined quite frequently in the study of the geometry of Banach spaces, see, e.g., [JL01, Chapter 1].

NN under general edit distance. For edit distance in general strings, the two known NN schemes with strongly sublinear query time achieve a constant factor approximation using n^{d^ϵ} storage for every fixed $\epsilon > 0$ [Ind04], or $2^{O(\sqrt{\log d \log\log d})}$ approximation using $(dn)^{O(1)}$ storage [OR07]. The latter result is obtained by embedding the corresponding metric into ℓ_1.

Chapter 6

External Applications of Product Spaces

In the previous chapter, we have shown how one can obtain new NN algorithms via embeddings into product spaces, yielding results that are impossible via embeddings into more classical spaces. We now show that embeddings into product spaces lead to improved algorithms for other applications, beyond the NN application.

We present two results based on embeddings into product spaces. The first result shows that we can estimate the edit distance between two strings of length d up to a $2^{\tilde{O}(\sqrt{\log d})}$ approximation factor in near-linear time. The second result constructs a sketching algorithm for the Ulam metric achieving constant approximation in poly-logarithmic space; furthermore, the sketch may be computed in the streaming model.

We now describe the results in more detail.

Computing edit distance in near-linear time. As previously mentioned, edit distance is of fundamental importance in a number of fields, such as computational biology (see more on edit distance in Section 2.1).

The basic problem on edit distance is to compute the edit distance between two strings of length d over some alphabet. The text-book dynamic programming runs in $O(d^2)$ time (see, e.g., [CLRS01] and references therein). This was only slightly improved by Masek and Paterson [MP80] to $O(d^2/\log^2 d)$ time for constant-size alphabets[1]. Their result from 1980 remains the best algorithm to this date.

Since near-quadratic time is too costly when working on large datasets, practitioners tend to rely on faster heuristics (see, e.g., [Gus97], [Nav01]). This leads to the question of finding fast algorithms with provable guarantees, specifically: can one *approximate* the edit distance between two strings in near-linear time [Ind01a, BEK$^+$03, BJKK04, BES06, CPSV00, Cor03, OR07, KN06, KR06]?

A linear-time $\sqrt{d}$-approximation algorithm immediately follows from the $O(d + R^2)$-time exact algorithm (see Landau, Myers, and Schmidt [LMS98]), where R is the edit distance between the input strings. Subsequent research improved the approximation first to $d^{3/7}$, and then to $d^{1/3+o(1)}$, due to, respectively, Bar-Yossef, Jayram, Krauthgamer, and Kumar [BJKK04], and Batu, Ergün, and Sahinalp [BES06].

[1] The result has been only recently extended to arbitrarily large alphabets by Bille and Farach-Colton [BFC08] with a $O(\log\log d)^2$ factor loss in time.

A *sublinear* time algorithm was obtained by Batu, Ergün, Kilian, Magen, Raskhodnikova, Rubinfeld, and Sami [BEK+03]. Their algorithm distinguishes the cases when the distance is $O(d^{1-\epsilon})$ vs. $\Omega(d)$ in $\tilde{O}(d^{1-2\epsilon} + d^{(1-\epsilon)/2})$ time for any $\epsilon > 0$. Note that their algorithm cannot distinguish distances, say, $O(d^{0.1})$ vs. $\Omega(d^{0.9})$.

On a related front, in 2005, the breakthrough result of Ostrovsky and Rabani gave an embedding of the edit distance metric into ℓ_1 with $2^{\tilde{O}(\sqrt{\log d})}$ distortion [OR07]. This result vastly improved related applications, namely NN and sketching. However, it did not have implications for computing edit distance between two strings in sub-quadratic time. In particular, to the best of our knowledge it is not known whether it is possible to compute their embedding in less than quadratic time.

Prior to our result, the best approximation remained the 2006 result of Batu, Ergün, and Sahinalp [BES06], achieving $d^{1/3+o(1)}$ approximation. Even for $d^{2-\epsilon}$ time, their approximation is $d^{\epsilon/3+o(1)}$.

Here, we show how to obtain $2^{\tilde{O}(\sqrt{\log d})}$ approximation in near-linear time. This is the first sub-polynomial approximation algorithm for computing the edit distance between two strings running in strongly subquadratic time. The main tool used to obtain our new algorithm is embeddings into *min*-product spaces (defined similarly to the max-product spaces).

Sketching algorithms for the Ulam distance. Our second application of product spaces ideas is to designing sketching and streaming algorithms for computing Ulam distance (see definition and motivation of the Ulam distance in Section 2.1). As defined in Section 1.3.2, the sketch of a point x is a (randomized) mapping of x into a short "fingerprint" $\mathbf{sk}(x)$ with the following property. For a pair of points x and y, the sketches $\mathbf{sk}(x)$ and $\mathbf{sk}(y)$ are sufficient to distinguish (with high probability) between the case when x, y are at distance $d(x, y) \leq R$, and the case when $d(x, y) > \alpha R$, for an approximation factor $\alpha > 1$ and a threshold parameter $R \in \mathbb{R}^+$. The main parameter of a sketching algorithm is its sketch size, the bit length of $\mathbf{sk}(x)$.

The sketching model is viewed as a basic computational primitive in massive data sets [AMS99, BBD+02, FM85]. For example, sketches of size s for an approximation α imply NN with space $n^{O(s)}$ with approximation α.

Here, we design a sketching algorithm for the Ulam metric that achieves $O(1)$-approximation using a sketch of size $\log^{O(1)} d$. This approximation is a significant improvement over the $O(\log d)$ approximation that follows immediately from the ℓ_1-embedding of Ulam metric [CK06]. Our sketching result is *tight* up to the exponent of the logarithm. Specifically, it is known that $O(1)$-approximation requires sketch size to be $\Omega(\log d \ / \ \log\log d)$ (see Chapter 7 and [AJP10]).

Furthermore, we show that our sketch is computable in the data stream model, thus answering an open question posed in [AJKS02, Section 6] on computing Ulam distance in a stream. Specifically, we can compute the sketch of a string P even if we have a sequential access to elements $P[1], P[2], \ldots P[d]$, using a total of polylog(d) space. To put our result in a perspective, for the problem of estimating the distance between two permutations interleaved in a stream, [AJKS02] give an $O(\sqrt{d} \log d)$-space streaming algorithm for a $1 + \epsilon$ approximation, when the distance is the *number of inversions* (Kendall tau distance).

Our sketching algorithm is based on the Ulam embeddings into product spaces from the previous chapter.

The results from this chapter have previously appeared in [AO09, AIK09].

6.1 Computing Edit Distance in Near-Linear Time

In this section, we design the following algorithm.

Theorem 6.1.1. *The edit distance between two strings $x, y \in \{0,1\}^d$ can be computed up to a factor of $2^{O(\sqrt{\log d \log\log d})}$ in $d \cdot 2^{O(\sqrt{\log d \log\log d})}$ time.*

This result immediately extends to two more related applications. The first application is to sublinear-time algorithms. In this scenario, the goal is to compute the distance between two strings x, y of the same length d in $o(d)$ time. For this problem, for any $\alpha < \beta \leq 1$, we can distinguish distance $O(d^\alpha)$ from distance $\Omega(d^\beta)$ in $O(d^{\alpha+2(1-\beta)+o(1)})$ time.

The second application is to the problem of pattern matching with errors. In this application, one is given a text T of length N and a pattern P of length d, and the goal is to report the substring of T that minimizes the edit distance to P. Our result immediately gives an algorithm for this problem running in $O(N \log N) \cdot 2^{\tilde{O}(\sqrt{\log d})}$ time with $2^{\tilde{O}(\sqrt{\log d})}$ approximation.

Before proceeding to the proof of Theorem 6.1.1, we will describe the general approach to prove our theorem, as well as review the Ostrovsky–Rabani embedding [OR07], which is also instrumental for our algorithm.

6.1.1 Proof overview

We will use the following two metrics. The first one is a variant of EMD over high-dimensional spaces. The second one is the (standard) notion of a graph metric.

We define *thresholded Earth-Mover Distance*, denoted TEMD_t for a fixed threshold $t > 0$, as the following distance on subsets A and B of size $s \in \mathbb{N}$ of some metric (M, d_M):

$$\mathrm{TEMD}_t(A, B) = \tfrac{1}{s} \min_{\tau: A \to B} \sum_{a \in A} \min\left\{\mathsf{d}_M(a, \tau(a)), t\right\} \tag{6.1}$$

where τ ranges over all bijections between sets A and B. TEMD_∞ is the simple Earth-Mover Distance (EMD), as defined in Section 2.1. We will always use $t = s$ and thus drop the subscript t; i.e., $\mathrm{TEMD} = \mathrm{TEMD}_s$.

A *graph (tree) metric* is a metric induced by a connected weighted graph (tree) G, where the distance between two vertices is the length of the shortest path between them. We denote by an arbitrary tree metric by TM. We also slightly abuse the notation by writing $\bigoplus_{\min}^k \mathrm{TM}$ to denote the min-product of k tree metrics (that could differ from each other).

We now describe the intuition behind our new algorithm. Our starting point is the Ostrovsky-Rabani embedding [OR07]. For strings x, y, as well as for all substrings σ of specific lengths, we compute some vectors v_σ living in low-dimensional ℓ_1 such that the distance between two such vectors approximates the edit distance between the associated (sub-)strings. In this respect, these vectors can be seen as an embedding of the considered strings into ℓ_1 of *polylogarithmic dimension*. Unlike the Ostrovsky-Rabani embedding, however, our embedding is *non-oblivious* in the sense that the vectors v_σ are computed given all the relevant strings σ. In contrast, Ostrovsky and Rabani give an *oblivious* embedding $\phi_d : \{0,1\}^d \to \ell_1$ such that $\|\phi_d(x) - \phi_d(y)\|_1$ approximates $\mathrm{ed}(x, y)$. However, the obliviousness comes at a high price: their embedding requires a high dimension, of order $\Omega(d)$, and a high computation time, of order $\Omega(d^2)$ (even when allowing randomized embedding, and a constant probability of a correctness). We further note that reducing the dimension of this embedding seems unlikely as suggested by the results on impossibility of dimensionality reduction within ℓ_1 [CS02, BC03, LN04]. Nevertheless, the general recursive approach of the Ostrovsky-Rabani embedding is the starting point of the algorithm from this paper.

The heart of our algorithm is a near-linear time algorithm that, given a sequence of low-dimensional vectors $v_1, \ldots v_d \in \ell_1$ and an integer $s < d$, constructs new vectors $q_1, \ldots q_m \in \ell_1^{O(\log^2 d)}$, where $m = d - s + 1$, with the following property. For all $i, j \in [m]$, the value $\|q_i - q_j\|_1$ approximates the TEMD distance between the sets $A_i = \{v_i, v_{i+1}, \ldots v_{i+s-1}\}$ and $A_j = \{v_j, v_{j+1}, \ldots v_{j+s-1}\}$. To accomplish this (non-oblivious) embedding, we proceed in two stages. First, we embed (obliviously) the TEMD metric into a *min-product of* ℓ_1*'s* of low dimension. In other words, for a set A, we associate a matrix $L(A)$, of polylogarithmic size, such that the TEMD distance between sets A and B is approximated by $\min_r \sum_t |L(A)_{rt} - L(B)_{rt}|$. Min-products help us simultaneously on two fronts: one is that we can apply a *weak* dimensionality reduction in ℓ_1, using the Cauchy projections, and the second one enables us to accomplish a low-dimensional TEMD embedding itself. Our embedding $L(\cdot)$ is not only low-dimensional, but it is also *linear*, allowing us to compute matrices $L(A_i)$ in near-linear time by performing one pass over the sequence $v_1, \ldots v_n$. Linearity is crucial here as even the total size of A_i's is $\sum_i |A_i| = (d - s + 1) \cdot s$, which can be as high as $\Omega(d^2)$, and so processing each A_i separately is infeasible.

In the second stage, we show how to embed a set of d points lying in a low-dimensional min-product of ℓ_1's back into a low-dimensional ℓ_1 with only small distortion. We note that this is not possible in general, with any bounded distortion, because such a set of points does not even form a metric. We show that this is possible when we assume that the semi-metric induced by the set of points approximates some metric (in our case, the set of points approximates the initial TEMD metric). The embedding from this stage starts by embedding a min-product of ℓ_1's into a low-dimensional min-product of tree metrics. We further embed the latter into an d-point metric supported by the shortest-path metric of a *sparse* graph. Finally, we observe that we can implement Bourgain's embedding on a sparse graph metric in *near-linear time*. These last two steps make our embedding non-oblivious.

6.1.2 Overview of the Ostrovsky–Rabani embedding

We now briefly describe the embedding of Ostrovsky and Rabani [OR07]. Some notions introduced here are used in our algorithm described in the next section.

The embedding of Ostrovsky and Rabani is recursive. For a fixed d, they construct the embedding of edit distance over strings of length d using the embedding of edit distance over strings of shorter lengths $l \leq d/2^{\sqrt{\log d \log\log d}}$. We denote their embedding of length-d strings by $\phi_d : \{0,1\}^d \to \ell_1$, and let $\mathsf{d}_d^{\mathrm{OR}}$ be the resulting distance: $\mathsf{d}_d^{\mathrm{OR}}(x, y) = \|\phi_d(x) - \phi_d(y)\|_1$. For two strings $x, y \in \{0,1\}^d$, the embedding is such that $\mathsf{d}_d^{\mathrm{OR}} = \|\phi_d(x) - \phi_d(y)\|_1$ approximates an "idealized" distance $\mathsf{d}_d^*(x, y)$, which itself approximates the edit distance between x and y.

Before describing the "idealized" distance d_d^*, we introduce some notation. Partition x into $b = 2^{\sqrt{\log d \log\log d}}$ blocks called $x^{(1)}, \ldots x^{(b)}$ of length $l = d/b$. Next, fix some $j \in [b]$ and $s \leq l$. We consider the set of all substrings of $x^{(j)}$ of length $l - s + 1$, embed each one recursively via ϕ_{l-s+1}, and define $S_j^s(x) \subset \ell_1$ to be the set of resulting vectors (note that $|S_j^s| = s$). Formally,

$$S_j^s(x) = \left\{\phi_{l-s+1}(x[(j-1)l + z : (j-1)l + z + l - s]) \mid z \in [s]\right\}.$$

Taking ϕ_{l-s+1} as given (and thus also the sets $S_j^s(x)$ for all x), define the new "idealized" distance d_d^* approximating the edit distance between strings $x, y \in \{0,1\}^d$ as

$$\mathsf{d}_d^*(x,y) = c \sum_{j=1}^{b} \sum_{\substack{f \in \mathbb{N} \\ s=2^f \le l}} \mathrm{TEMD}(S_j^s(x), S_j^s(y)) \tag{6.2}$$

where TEMD is the thresholded Earth-Mover Distance (defined in Eqn. (6.1)), and c is a sufficiently large normalization constant ($c \ge 12$ suffices). Using the terminology from the preliminaries, the distance function d_d^* can be viewed as the distance function of the sum-product of TEMDs, i.e., $\bigoplus_{\ell_1}^{b} \bigoplus_{\ell_1}^{O(\log d)} \mathrm{TEMD}$, and the embedding into this product space is attained by the natural identity map (on sets S_j^s).

The key idea is that the distance $\mathsf{d}_d^*(x,y)$ approximates edit distance well, assuming that ϕ_{l-s+1} approximates edit distance well, for all $s = 2^f$ where $f \in \{1, 2, \ldots \lfloor \log_2 l \rfloor\}$. Formally, Ostrovsky and Rabani show that:

Fact 6.1.2 ([OR07])**.** *Fix d and $b < d$, and let $l = d/b$. Let $D_{d/b}$ be an upper bound on distortion of ϕ_{l-s+1} viewed as an embedding of edit distance on strings $\{x[i : i+l-s], y[i : i+l-s] \mid i \in [d-l+s]\}$, for all $s = 2^f$ where $f \in \{1, 2, \ldots \lfloor \log_2 l \rfloor\}$. Then,*

$$\mathrm{ed}(x,y) \le \mathsf{d}_d^*(x,y) \le \mathrm{ed}(x,y) \cdot \big(D_{d/b} + b\big) \cdot O(\log d).$$

To obtain a complete embedding, it remains to construct an embedding approximating d_d^* up to a small factor. In fact, if one manages to approximate d_d^* up to a poly-logarithmic factor, then the final distortion comes out to be $2^{O(\sqrt{\log d \log\log d})}$. This follows from the following recurrence on the distortion factor D_d. Suppose ϕ_d is an embedding that approximates d_d^* up to a factor $\log^{O(1)} d$. Then, if D_d is the distortion of ϕ_d (as an embedding of edit distance), then Fact 6.1.2 immediately implies that, for $b = 2^{\sqrt{\log d \log\log d}}$,

$$D_d \le D_{d/2^{\sqrt{\log d \log\log d}}} \cdot \log^{O(1)} d + 2^{O(\sqrt{\log d \log\log d})}.$$

This recurrence solves to $D_d \le 2^{O(\sqrt{\log d \log\log d})}$ as proven in [OR07].

Concluding, to complete a step of the recursion, it is sufficient to embed the metric given by d_d^* into ℓ_1 with a polylogarithmic distortion. Recall that d_d^* is the distance of the metric $\bigoplus_{\ell_1}^{b} \bigoplus_{\ell_1}^{O(\log d)} \mathrm{TEMD}$, and thus, one just needs to embed TEMD into ℓ_1. Indeed, Ostrovsky and Rabani show how to embed a relaxed (but sufficient) version of TEMD into ℓ_1 with $O(\log d)$ distortion, yielding the desired embedding ϕ_d, which approximates d_d^* up to a $O(\log d)$ factor at each level of recursion. We note that the required dimension is $\tilde{O}(d)$.

6.1.3 Main algorithm: proof of Theorem 6.1.1

We now describe our general approach. Fix $x \in \{0,1\}^d$. For each substring σ of x, we construct a low-dimensional vector v_σ such that, for any two substrings σ, τ of the same length, the edit distance between σ and τ is approximated by the ℓ_1 distance between the vectors v_σ and v_τ. We note that the embedding is non-oblivious: to construct vectors v_σ we need to know *all* the substrings of x in advance (akin to Bourgain's embedding guarantee). We also note that computing such vectors is enough to solve the problem of approximating the edit distance between two strings, x and y. Specifically, we apply this procedure to the string $x' = x \circ y$, the concatenation of x and y, and then compute the ℓ_1 distance between the vectors corresponding to x and y, substrings of x'.

More precisely, for each length $m \in W$, for some set $W \subset [d]$ specified later, and for each substring $x[i : i+m-1]$, where $i = 1, \dots d-m+1$, we compute a vector $v_i^{(m)}$ in ℓ_1^α, where $\alpha = 2^{\tilde{O}(\sqrt{\log d})}$. The construction is inductive: to compute vectors $v_i^{(m)}$, we use vectors $v_i^{(l)}$ for $l \ll m$ and $l \in W$. The general approach of our construction is based on the analysis of the recursive step of Ostrovsky and Rabani, described in Section 6.1.2. In particular, our vectors $v_i^{(m)} \in \ell_1$ will also approximate the d_m^* distance (given in Eqn. (6.2)) with sets S_i^s defined using vectors $v_i^{(l)}$ with $l \ll m$.

The main challenge is to process one level (vectors $v_i^{(m)}$ for a fixed m) in near-linear time. Besides the computation time itself, a fundamental difficulty in applying the approach of Ostrovsky and Rabani directly is that their embedding would give a much higher dimension α, proportional to $\tilde{O}(m)$. Thus, if we were to use their embedding, even storing all the vectors would take quadratic space.

To overcome this last difficulty, we settle on non-obliviously embedding the set of substrings $x[i : i+m-1]$ for $i \in [d-m+1]$ under the "ideal" distance d_m^* with $\log^{O(1)} d$ distortion (formally, under the distance d_m^* from Eqn. (6.2), when $S_j^s(x[i : i+m-1]) = \left\{v_{i+(j-1)l+z-1}^{(l-s+1)} \mid z \in [s]\right\}$ for $l = m/2^{\sqrt{\log d \log\log d}}$). Existentially, we know that there exist vectors $v_i^{(m)} \in \mathbb{R}^{O(\log^2 d)}$, such that $\|v_i^{(m)} - v_j^{(m)}\|_1$ approximate $\mathsf{d}_m^*(x[i : i+m-1], x[j : j+m-1])$ for all i, j — this follows by the standard Bourgain's embedding [Bou85]. We show that we can also compute these $v_i^{(m)}$'s efficiently for all $i \in [d-m+1]$, albeit with an additional polylogarithmic loss in approximation.

The main building block is the following theorem. It shows how to approximate the TEMD distance for the desired sets S_j^s.

Theorem 6.1.3. *Let $n \in \mathbb{N}$ and $s \in [n]$. Let $v_1, \dots v_n$ be vectors in $\{-M, \dots M\}^\alpha$, where $M = n^{O(1)}$ and $\alpha \le n$. Define sets $A_i = \{v_i, v_{i+1}, \dots v_{i+s-1}\}$ for $i \in [n-s+1]$.*

Let $t = O(\log^2 n)$. We can compute (randomized) vectors $q_i \in \ell_1^t$ for $i \in [n-s+1]$ such that for any $i, j \in [n-s+1]$, with high probability, we have

$$\mathrm{TEMD}(A_i, A_j) \le \|q_i - q_j\|_1 \le \mathrm{TEMD}(A_i, A_j) \cdot \log^{O(1)} n.$$

Furthermore, computing all vectors q_i takes $\tilde{O}(n\alpha)$ time.

To map the statement of this theorem to the above description, we mention that, for each $l = m/b$ for $m \in W$, we apply the theorem to vectors $\left(v_i^{(l-s+1)}\right)_{i \in [d-l+s]}$ for each $s = 1, 2, 4, 8, \dots 2^{\lfloor \log_2 l \rfloor}$, setting $n = d - l + s$.

We prove Theorem 6.1.3 in later sections. Once we have Theorem 6.1.3, it becomes relatively straight-forward (albeit a bit technical) to prove the main theorem, Theorem 6.1.1. We complete the proof of Theorem 6.1.1 next, assuming Theorem 6.1.3.

Proof of Theorem 6.1.1. We start by appending y to the end of x; we will work with the new version of x only. Let $b = 2^{\sqrt{\log d \log\log d}}$ and $\alpha = O(b \log^3 d)$. We construct vectors $v_i^{(m)} \in \mathbb{R}^\alpha$ for $m \in W$, where $W \subset [d]$ is a carefully chosen set of size $2^{O(\sqrt{\log d \log\log d})}$. Namely, W is the minimal set such that: $d \in W$, and, for each $i \in W$ with $i \ge b$, we have that $i/b - 2^j + 1 \in W$ for all integers $j \le \lfloor \log_2 i/b \rfloor$. It is easy to show by induction that the size of W is $2^{O(\sqrt{\log d \log\log d})}$.

Fix an $m \in W$ such that $m \le b^2 = 2^{2\sqrt{\log d \log\log d}}$. We define the vector $v_i^{(m)}$ to be equal to $h_m(x[i : i+m-1])$, where $h_m : \{0,1\}^m \to \{0,1\}^\alpha$ is a randomly chosen function. It is readily seen that $\|v_i^{(m)} - v_j^{(m)}\|_1$ approximates $\mathrm{ed}(x[i : i+m-1], x[j : j+m-1])$ up to $b^2 = 2^{2\sqrt{\log d \log\log d}}$ approximation factor, for each $i, j \in [d-m+1]$.

Now consider $m \in W$ such that $m > b^2$. Let $l = m/b$. First we construct vectors approximating TEMD on sets $A_i^{m,s} = \left\{ v_{i+z}^{(l-s+1)} \mid z = 0, \dots s-1 \right\}$, where $s = 1, 2, 4, 8, \dots, l$ and $i \in [d-l+s]$. In particular, for a fixed $s \in [l]$ equal to a power of 2, we apply Theorem 6.1.3 to the set of vectors $\left(v_i^{(l-s+1)}\right)_{i \in [d-l+s]}$ obtaining vectors $\left(q_i^{(m,s)}\right)_{i \in [d-l+1]}$. Theorem 6.1.3 guarantees that, for each $i, j \in [d-l+1]$, the value $\|q_i^{(m,s)} - q_j^{(m,s)}\|_1$ approximates $\mathrm{TEMD}(A_i^{m,s}, A_j^{m,s})$ up to a factor of $\log^{O(1)} d$. We can then use these vectors $q_i^{(m,s)}$ to obtain the vectors $v_i^{(m)} \in \mathbb{R}^\alpha$ that approximate the "idealized" distance d_m^* on substrings $x[i : i+m-1]$, for $i \in [d-m+1]$. Specifically, we let the vector $v_i^{(m)}$ be a concatenation of vectors $q_{i+(j-1)l}^{(m,s)}$, where $j \in [b]$, and s goes over all powers of 2 less than l:

$$v_i^{(m)} = \left(q_{i+(j-1)l}^{(m,s)} \right)_{\substack{j \in [b] \\ s = 2^f \leq l, f \in \mathbb{N}}}.$$

Then, the vectors $v_i^{(m)}$ approximate the distance d_m^* (given in Eqn. (6.2)) up to a $\log^{O(1)} d$ approximation factor, with the sets $S_j^s(x[i : i+m-1])$ taken as

$$S_j^s(x[i : i+m-1]) = A_{i+(j-1)l}^{m,s} = \left\{ v_{i+(j-1)l+z}^{(l-s+1)} \mid z = 0, \dots s-1 \right\},$$

for $i \in [d-m+1]$ and $j \in [b]$.

The algorithm finishes by outputting $\|v_1^{(d)} - v_{d+1}^{(d)}\|$, which is an approximation to the edit distance between $x[1:d]$ and $x[d+1:2d] = y$. The total running time is $O(|W| \cdot d \cdot b^{O(1)} \cdot \log^{O(1)} d) = d \cdot 2^{O(\sqrt{\log d \log\log d})}$.

It remains to analyze the resulting approximation. Let D_m be the approximation achieved by vectors $v_i^{(k)} \in \ell_1$ for substrings of x of lengths k, where $k \in W$ and $k \leq m$. Then, using Fact 6.1.2 and the fact that vectors $v_i^{(m)} \in \ell_1$ approximate d_m^*, we have that

$$D_m \leq \log^{O(1)} d \cdot \left(D_{m/b} + 2^{\sqrt{\log d \log\log d}} \right).$$

Since the total number of recursion levels is bounded by $\log_b d = \sqrt{\frac{\log d}{\log\log d}}$, we deduce that $D_d = 2^{O(\sqrt{\log d \log\log d})}$. □

6.1.4 Proof of Theorem 6.1.3

The proof of Theorem 6.1.3 proceeds in two stages. In the first stage we show an embedding of the TEMD metric into a low-dimensional space. Specifically, we show an (oblivious) embedding of TEMD into a *min-product* of ℓ_1. Recall that the min-product of ℓ_1, denoted $\bigoplus_{\min}^l \ell_1^k$, is a semi-metric where the distance between two l-by-k vectors $x, y \in \mathbb{R}^{l \times k}$ is $\mathsf{d}_{\min,1}(x,y) = \min_{i \in [l]} \left\{ \sum_{j \in [k]} |x_{i,j} - y_{i,j}| \right\}$. Our min-product of ℓ_1's has dimensions $l = O(\log n)$ and $k = O(\log^3 n)$. The min-product can be seen as helping us on two fronts: one is the embedding of TEMD into ℓ_1 (of initially high-dimension), and another is a *weak* dimensionality reduction in ℓ_1, using Cauchy projections. Both of these embeddings are of the following form: consider a randomized embedding f into (standard) ℓ_1 that has no contraction (w.h.p.) but the expansion is bounded only in the expectation (as opposed to w.h.p.). To obtain a "w.h.p." expansion, one standard approach is to sample f many times and concentrate the expectation. This approach, however, will necessitate a high number of samples of f, and thus yield a high final dimension. Instead, the min-product allows us to take only $O(\log n)$ independent samples of f.

We note that our embedding of TEMD into min-product of ℓ_1, denoted λ, is linear in the sets A: $\lambda(A) = \sum_{a \in A} \lambda(\{a\})$. The linearity allows us to compute the embedding of sets A_i in a streaming fashion: the embedding of A_{i+1} is obtained from the embedding of A_i with $\log^{O(1)} n$ additional processing.

In the second stage, we show that, given a set of n points in min-product of ℓ_1's, we can (non-obliviously) embed these points into low-dimensional ℓ_1 with $O(\log n)$ distortion. The time required is near-linear in n and the dimensions of the min-product of ℓ_1's.

To accomplish this step, we start by embedding the min-product of ℓ_1's into a min-product of tree metrics. Next, we show that n points in the low-dimensional min-product of tree metrics can be embedded into a graph metric supported by a *sparse* graph. We note that this is in general not possible, with any (even non-constant) distortion. We show that this is possible when we assume that our subset of the min-product of tree metrics approximates some actual metric (in our case, the min-product approximates the TEMD metric). Finally, we observe that we can implement Bourgain's embedding in near-linear time on a sparse graph metric.

We then conclude with the proof of Theorem 6.1.3.

Embedding EMD into min-product of ℓ_1

In the next lemma, we show how to embed TEMD into a min-product of ℓ_1's of low dimension. Moreover, when the sets A_i are obtained from a sequence of vectors $v_1, \ldots v_n$, by taking $A_i = \{v_i, \ldots v_{i+s-1}\}$, we can compute the embedding in near-linear time.

Lemma 6.1.4. *Fix $n, M \in \mathbb{N}$ and $s \in [n]$. Suppose we have n vectors $v_1, \ldots v_n$ in $\{-M, \ldots M\}^\alpha$ for some $\alpha \leq n$. Consider the sets $A_i = \{v_i, v_{i+1}, \ldots v_{i+s-1}\}$, for $i \in [n-s+1]$.*

Let $k = O(\log^3 n)$. We can compute (randomized) vectors $q_i \in \ell_1^k$ for $i \in [n-s+1]$ such that, for any $i, j \in [n-s+1]$ we have that

- $\Pr\left[\|q_i - q_j\|_1 \leq \mathrm{TEMD}(A_i, A_j) \cdot O(\log^2 n)\right] \geq 0.1$ *and*
- $\|q_i - q_j\|_1 \geq \mathrm{TEMD}(A_i, A_j)$ *w.h.p.*

The computation time is $\tilde{O}(n\alpha)$.

Thus, we can embed the TEMD *metric over sets A_i into $\bigoplus_{\min}^{l} \ell_1^k$, for $l = O(\log n)$, such that the distortion is $O(\log^2 n)$ w.h.p. The computation time is $\tilde{O}(n\alpha)$.*

Proof. First, we show how to embed TEMD metric over the sets A_i into ℓ_1 of dimension $M^{O(\alpha)} \cdot O(\log n)$. For this purpose, we use a slight modification of the embedding of [AIK08] (it can also be seen as a strengthening of the TEMD embedding of Ostrovsky and Rabani).

The embedding of [AIK08] constructs $m = O(\log s)$ embeddings ψ_i, each of dimension $h = M^{O(\alpha)}$, and then the final embedding is just the concatenation $\psi = \psi_1 \circ \psi_2 \ldots \circ \psi_m$. For $i = 1, \ldots m$, we impose a randomly shifted grid of side-length $R_i = 2^{i-2}$. Then ψ_i has a coordinate for each cell and the value of that coordinate, for a set A, equals the number of points from A falling into the corresponding cell times R_i. Now, if we scale ψ up by

$\Theta(\frac{1}{s}\log n)$, Theorem 3.1 from [AIK08] says that the vectors $q'_i = \psi(A_i)$ satisfy the condition that, for any $i, j \in [n-s+1]$, we have:

- $\mathbb{E}\left[\|q'_i - q'_j\|_1\right] \leq \mathrm{TEMD}(A_i, A_j) \cdot O(\log^2 n)$ and
- $\|q'_i - q'_j\|_1 \geq \mathrm{TEMD}(A_i, A_j)$ w.h.p.

Thus, the vectors q'_i satisfy the promised properties except they have a high dimension.

To reduce the dimension of q'_i's, we apply a weak ℓ_1 dimensionality reduction via 1-stable (Cauchy) projections. Namely, we pick a random matrix P of size $k = O(\log^3 n)$ by mh, the dimension of ψ, where each entry is distributed according to the Cauchy distribution, which has probability distribution function $f(x) = \frac{1}{\pi} \cdot \frac{1}{1+x^2}$. Now define $q_i = P \cdot q'_i \in \ell_1^k$. Standard properties of the ℓ_1 dimensionality reduction guarantee that the vectors q_i satisfy the properties promised in the lemma statement, after an appropriate rescaling (see Theorem 5 of [Ind06] with $\epsilon = 1/2$, $\gamma = 1/6$, and $\delta = n^{-O(1)}$).

It remains to show that we can compute the vectors q_i in $\tilde{O}(n\alpha)$ time. For this, we note that the resulting embedding $P \cdot \psi(A)$ is linear, namely $P \cdot \psi(A) = \sum_{a \in A} P \cdot \psi(\{a\})$. Thus, we can use the idea of a sliding window over the stream $v_1, \dots v_n$ to compute $q_i = P \cdot \psi(A_i)$ iteratively. Specifically, note that

$$q_{i+1} = P \cdot \psi(A_{i+1}) = P \cdot \psi(A_i \cup \{v_{i+s}\} \setminus \{v_i\}) = q_i + P \cdot \psi(\{v_{i+s}\}) - P \cdot \psi(\{v_i\}).$$

Since we can compute $P \cdot \psi(\{v_i\})$, for any i, in $\alpha \cdot \log^{O(1)} n$ time, we conclude that the total time to compute q_i's is $O(n\alpha \cdot \log^{O(1)} n)$.

Finally, we show how we obtain an efficient embedding of TEMD into min-product of ℓ_1's.

We apply the above procedure $l = O(\log n)$ times. Let $q_i^{(z)}$ be the resulting vectors, for $i \in [n-s+1]$ and $z \in [l]$. The embedding of a set A_i is the concatenation of the vectors $q_i^{(z)}$, namely $Q_i = (q_i^{(1)}, q_i^{(2)}, \dots q_i^{(l)}) \in \bigoplus_{\min}^l \ell_1^k$. The Chernoff bound implies that w.h.p., for any $i, j \in [n-s+1]$, we have that

$$\mathsf{d}_{\min,1}(Q_i, Q_j) = \min_{z \in [l]} \|q_i^{(z)} - q_j^{(z)}\| \leq \mathrm{TEMD}_s(A_i, A_j) \cdot O(\log^2 n).$$

Also, $\mathsf{d}_{\min,1}(Q_i, Q_j) \geq \mathrm{TEMD}_s(A_i, A_j)$ w.h.p. trivially. Thus the vectors Q_i are an embedding of the TEMD metric on A_i's into $\bigoplus_{\min}^l \ell_1^k$ with distortion $O(\log^2 n)$ w.h.p. □

Embedding of min-product of ℓ_1 into low-dimensional ℓ_1

In this section, we show that n points $Q_1, \dots Q_n$ in the semi-metric space $\bigoplus_{\min}^l \ell_1^k$ can be embedded into ℓ_1 of dimension $O(\log^2 n)$ with distortion $\log^{O(1)} n$. The embedding works under the assumption that the semi-metric on $Q_1, \dots Q_n$ is a $\log^{O(1)} n$ approximation of some metric. We start by showing that we can embed a min-product of ℓ_1's into a min-product of tree metrics.

Lemma 6.1.5. *Fix $n, M \in \mathbb{N}$ such that $M = n^{O(1)}$. Consider n vectors $v_1, \dots v_n$ in $\bigoplus_{\min}^l \ell_1^k$, for some $l, k \in \mathbb{N}$, where each coordinate of each v_i lies in the set $\{-M, \dots, M\}$. We can embed these vectors into a min-product of $O(l \cdot \log^2 n)$ tree metrics, i.e., $\bigoplus_{\min}^{O(l \log^2 n)} \mathrm{TM}$, incurring distortion $O(\log n)$ w.h.p. The computation time is $\tilde{O}(n \cdot kl)$.*

Proof. We consider all thresholds 2^t, for $t \in \{0, 1, \ldots, \log M\}$. For each threshold 2^t, and for each coordinate of the min-product (i.e., ℓ_1^k), we create $O(\log n)$ tree metrics. Each tree metric is independently created as follows. We again use randomly shifted grids. Specifically, we define a hash function $h : \ell_1^k \to \mathbb{Z}^k$ as

$$h(x_1, \ldots, x_k) = \left(\left\lfloor \frac{x_1 + u_1}{2^t} \right\rfloor, \left\lfloor \frac{x_2 + u_2}{2^t} \right\rfloor, \ldots, \left\lfloor \frac{x_k + u_k}{2^t} \right\rfloor \right),$$

where each u_t is chosen at random from $[0, 2^t)$. We create each tree metric so that the nodes corresponding to the points hashed by h to the same value are at distance 2^t (this creates a set of stars), and each pair of points that are hashed to different values are at distance $2Mk$ (we connect the roots of the stars). It is easy to verify that for two points $x, y \in \ell_1^k$, the following holds

$$1 - \frac{\|x - y\|_1}{2^t} \leq \Pr_h[h(x) = h(y)] \leq e^{-\|x-y\|_1/2^t}.$$

By the Chernoff bound, if $x, y \in \ell_1^k$ are at distance at most 2^t for some t, they will be at distance at most 2^{t+1} in one of the tree metrics with high probability.

On the other hand, let v_i and v_j be two input vectors at distance greater than 2^t. The probability that they are at distance smaller than $2^t/c \log n$ in any of the $O(\log^2 n)$ tree metrics, is at most n^{-c+1} for any $c > 0$, by union bound.

Therefore, we multiply the weights of all edges in all trees by $O(\log n)$ to achieve a proper (non-contracting) embedding. □

We now show that we can embed a subset of the min-product of tree metrics into a graph metric, assuming the subset is close to a metric.

Lemma 6.1.6. *Consider a semi-metric $\mathcal{M} = (X, \xi)$ of size n in $\bigoplus_{\min}^{l} \mathrm{TM}$ for some $l \in \mathbb{N}$, where each tree metric in the product is of size $O(n)$. Suppose $\mathcal{M}$ is a γ-near metric (i.e., it is embeddable into a metric with γ distortion). Then we can embed $\mathcal{M}$ into a connected weighted graph with $O(nl)$ edges with distortion γ in $O(nl)$ time.*

Proof. We consider l separate trees each on $O(n)$ nodes, corresponding to each of l dimensions of the min-product. We identify the nodes of trees that correspond to the same point in the min-product, and collapse them into a single node. The graph we obtain has at most $O(nl)$ edges. Denote the shortest-path metric it spans with $\mathcal{M}' = (V, \rho)$, and denote our embedding with $\phi : X \to V$. Clearly, for each pair u, v of points in X, we have $\rho(\phi(u), \phi(v)) \leq \xi(u, v)$. If the distance between two points shrinks after embedding, then there is a sequence of points $w_0 = u$, w_1, $\ldots$, w_{k-1}, $w_k = v$ such that $\rho(\phi(u), \phi(v)) = \xi(w_0, w_1) + \xi(w_1, w_2) + \cdots + \xi(w_{k-1}, w_k)$. Because $\mathcal{M}$ is a γ-near metric, there exists a metric $\xi^\star : X \times X \to [0, \infty)$, such that $\xi^\star(x, y) \leq \xi(x, y) \leq \gamma \cdot \xi^\star(x, y)$, for all $x, y \in X$. Therefore,

$$\rho(\phi(u), \phi(v)) = \sum_{i=0}^{k-1} \xi(w_i, w_{i+1}) \geq \sum_{i=0}^{k-1} \xi^\star(w_i, w_{i+1}) \geq \xi^\star(w_0, w_k) = \xi^\star(u, v) \geq \xi(u, v)/\gamma.$$

Hence, it suffices to multiply all edge weights of the graph by γ to achieve a non-contractive embedding. Since there was no expansion before, it is now bounded by γ. □

We now show how to embed the shortest-path metric of a graph into a low dimensional ℓ_1-space in time near-linear in the graph size. For this purpose, we implement Bourgain's embedding [Bou85] in near-linear time. We use the following version of Bourgain's embedding, which follows from the analysis in [Mat02].

Lemma 6.1.7 (Bourgain's embedding [Mat02]). *Let $\mathcal{M} = (X, \rho)$ be a finite metric on n points. There is an algorithm that computes an embedding $f : X \to \ell_1^t$ of $\mathcal{M}$ into ℓ_1^t for $t = O(\log^2 n)$ such that, with high probability, for each $u, v \in X$, we have $\rho(u,v) \le \|f(u) - f(v)\|_1 \le \rho(u,v) \cdot O(\log n)$.*

Specifically, for coordinate $i \in [k]$ of f, the embedding associates a nonempty set $A_i \subseteq X$ such that $f(u)_i = \rho(u, A_i) = \min_{a \in A} \rho(u,a)$. Each A_i is samplable in linear time.

The running time of the algorithm is $O(g(n) \cdot \log^2 n)$, where $g(n)$ is the time necessary to compute the distance of all points to a given fixed subset of points.

Lemma 6.1.8. *Consider a connected graph $G = (V, E)$ on n nodes with m edges and a weight function $w : E \to [0, \infty)$. There is a randomized algorithm that embeds the shortest path metric of G into $\ell_1^{O(\log^2 n)}$ with $O(\log n)$ distortion, with high probability, in $O(m \log^3 n)$ time.*

Proof. Let $\psi : V \to \ell_1^{O(\log^2 n)}$ be the embedding given by Lemma 6.1.7. For any nonempty subset $A \subseteq V$, we can compute $\rho(v, A)$ for all $v \in V$ by Dijkstra's algorithm in $O(m \log n)$ time. The total running time is thus $O(m \log^3 n)$. $\square$

Finalization of the proof of Theorem 6.1.3

We first apply Lemma 6.1.4 to embed the sets A_i into $\bigoplus_{\min}^{O(\log n)} \ell_1^k$ with distortion at most $O(\log^2 n)$ with high probability, where $k = O(\log^3 n)$. We write v_i, $i \in [n-s+1]$, to denote the embedding of A_i. Note that the TEMD distance between two different A_i's is at least $1/s \ge 1/n$, and so is the distance between two different v_i's. We multiply all coordinates of v_i's by $2kn = \tilde{O}(n)$ and round them to the nearest integer. This way we obtain vectors v_i' with integer coordinates in $\{-2knM - 1, \ldots, 2knM + 1\}$. Consider two vectors v_i and v_j. Let D be their distance, and let D' be the distance between the corresponding v_i' and v_j'. We claim that $knD \le D' \le 3knD$, and it suffices to show this claim for $v_i \ne v_j$, in which case we know that $D \ge 1/n$. Each coordinate of the min-product is ℓ_1^k, and we know that in each of the coordinates the distance is at least D. Consider a given coordinate of the min-product, and let d and d' be the distance before and after the scaling and rounding, respectively. On the one hand,

$$\frac{d'}{d} \ge \frac{2knd - k}{d} \ge 2kn - \frac{k}{D} \ge 2kn - kn = kn,$$

and on the other,

$$\frac{d'}{d} \le \frac{2knd + k}{d} \le 2kn + \frac{k}{D} \le 2kn + kn = 3kn.$$

Therefore, in each coordinate, the distance gets scaled by a factor in the range $[kn, 3kn]$. We now apply Lemma 6.1.5 to v_i''s and obtain their embedding into a min-product of tree metrics. Then, we divide all distances in the trees by kn, and achieve an embedding of v_i's into a min-product of trees with distortion at most 3 times larger than that implied by Lemma 6.1.5, which is $O(\log n)$.

The resulting min-product of tree metrics need not be a metric, but it is a γ-near metric, where $\gamma = O(\log^3 n)$ is the expansion incurred so far. We therefore embed the min-product of tree metrics into the shortest-path metric of a weighted graph by using Lemma 6.1.6 with expansion at most γ. Finally, we embed this metric into a low dimensional ℓ_1 metric space with distortion $O(\log^2 n)$ by using Lemma 6.1.8.

6.1.5 Extensions

We now present two extensions of our main algorithm: sublinear-time approximation of edit distance, and approximate pattern matching under edit distance.

Sublinear-time approximation. We now present a sublinear-time algorithm for distinguishing pairs of strings with small edit distance from pairs with large edit distance. Let x and y be the two strings. The algorithm partitions them into blocks $\widetilde{x}_i$ and $\widetilde{y}_i$ of the same length such that $x = \widetilde{x}_1 \ldots \widetilde{x}_b$ and $y = \widetilde{y}_1 \ldots \widetilde{y}_b$. Then it compares $\widetilde{x}_i$ to $\widetilde{y}_i$ for a number of random i. If it finds a very different pair of blocks $\widetilde{x}_i$ to $\widetilde{y}_i$, the distance between x and y is large. Otherwise, the edit distance between x and y is likely to be small. Our edit distance algorithm is used for approximating the distance between specific $\widetilde{x}_i$ and $\widetilde{y}_i$.

Theorem 6.1.9. *Let α and β be two constants such that $0 \le \alpha < \beta \le 1$. There is an algorithm that distinguishes pairs of strings with edit distance $O(d^\alpha)$ from those with distance $\Omega(d^\beta)$ in time $d^{\alpha+2(1-\beta)+o(1)}$.*

Proof. Let $f(d) = 2^{O(\sqrt{\log d \log\log d})}$ be a non-decreasing function that bounds the approximation factor of the algorithm given by Theorem 6.1.1. Let $b = \frac{d^{\beta-\alpha}}{f(d)\cdot\log d}$. We partition the input strings x and y into b blocks, denoted $\widetilde{x}_i$ and $\widetilde{y}_i$ for $i \in [b]$, of length d/b each.

If $\mathrm{ed}(x,y) = O(d^\alpha)$, then $\max_i \mathrm{ed}(\widetilde{x}_i,\widetilde{y}_i) \le \mathrm{ed}(x,y) = O(d^\alpha)$. On the other hand, if $\mathrm{ed}(x,y) = \Omega(d^\beta)$, then $\max_i \mathrm{ed}(\widetilde{x}_i,\widetilde{y}_i) \ge \mathrm{ed}(x,y)/b = \Omega(d^\alpha \cdot f(d) \cdot \log d)$. Moreover, the number of blocks i such that $\mathrm{ed}(\widetilde{x}_i,\widetilde{y}_i) \ge \mathrm{ed}(x,y)/2b = \Omega(d^\alpha \cdot f(d) \cdot \log d)$ is at least

$$\frac{\mathrm{ed}(x,y) - b \cdot \mathrm{ed}(x,y)/2b}{d/b} = \Omega(d^{\beta-1} \cdot b).$$

Therefore, we can tell the two cases apart with constant probability by sampling $O(d^{1-\beta})$ pairs of blocks $(\widetilde{x}_i,\widetilde{y}_i)$ and checking if any of the pairs is at distance $\Omega(d^\alpha \cdot f(d) \cdot \log d)$. Since for each such pair of strings, we only have to tell edit distance $O(d^\alpha)$ from $\Omega(d^\alpha \cdot f(d) \cdot \log d)$, we can use the algorithm of Theorem 6.1.1. We amplify the probability of success of that algorithm in the standard way by running it $O(\log d)$ times. The total running time of the algorithm is $O(d^{1-\beta}) \cdot O(\log d) \cdot (d/b)^{1+o(1)} = O(d^{\alpha+2(1-\beta)+o(1)})$. $\square$

Pattern matching. Our algorithm can be used for approximating the edit distance between a pattern P of length d and all length-d substrings of a text T. Let $N = |T|$. For every $s \in [N - 2d + 1]$ of the form $id + 1$, we concatenate T's length-$2d$ substring that starts at index s with P, and compute an embedding of edit distance between all length-d substrings of the newly created string into ℓ_1^α for $\alpha = 2^{O(\sqrt{\log d \log\log d})}$. We routinely amplify the probability of success of each execution of the algorithm by running it for $O(\log N)$ times and selecting the median of the returned values. The running time of the algorithm is $O(N \log N) \cdot 2^{O(\sqrt{\log d \log\log d})}$.

The distance between each of the substrings and the pattern is approximate up to a factor of $2^{O(\sqrt{\log d \log\log d})}$, and can be used both for finding approximate occurrences of P in T, and for finding a substring of T that is approximately closest to P.

6.2 Sketching and Streaming for Ulam Distance

In this section, we describing our sketching and streaming algorithms for the Ulam distance, summarized in the following theorem.

Theorem 6.2.1. *We can compute a randomized sketch* $\mathbf{sk} : \mathrm{Ulam}_d \to \{0,1\}^s$ *for* $s = (\log d)^{O(1)}$ *in the streaming model, with* $(\log d)^{O(1)}$ *time per input character, with the following property. For every* $P, Q \in \mathrm{Ulam}_d$*, with high probability, given* $\mathbf{sk}(P)$ *and* $\mathbf{sk}(Q)$ *only, one can approximate* $\mathrm{ed}(P,Q)$ *within a constant approximation.*

As mentioned before, the main ingredient here is the embedding of Ulam distance from Theorem 5.2.1. However, to achieve a sketching algorithm, we use two more tools: subsampling (i.e., projecting a vector on a random subset of coordinates) and sketching of *heavy hitters* [CM05a] (which enable the recovery of coordinates on which the two sketched vectors differ considerably). This idea is somewhat related to the L_k norm estimation algorithm of [IW05], although the technical development is very different here. We do not provide a sketch of the space $\bigoplus_{(\ell_p)^p} \bigoplus_{\ell_\infty} \ell_1$ in its full generality, and in fact, very recently, it has been proven that this space does not admit short sketches when $p > 1$ [JW09]. Instead, we make use of additional properties of our embedding's images. Finally, to obtain a data stream algorithm for computing the sketch, we employ the *block* heavy hitters algorithm of [ADI08], as well as a technique of an attenuated window in the stream.

6.2.1 A simple sketching algorithm

We start by giving a simpler construction of a polylog-space sketching algorithm Ulam_d. While this sketch is not computable in a stream as is, we will extend the algorithm from here to obtain a streamable algorithm in Section 6.2.2.

Theorem 6.2.2. *There exists a sketching algorithm for* Ulam_d *achieving constant approximation using* $\log^{O(1)} d$ *space.*

Proof. Let $P, Q \in \mathrm{Ulam}_d$. We use the notation from Section 5.2.1. Notably, let $\varphi, \varphi_a, \varphi_{a,k}$ be as defined in Section 5.2.1, and let $\zeta, \zeta_a, \zeta_{a,k}$ be respectively $\varphi(P) - \varphi(Q), \varphi_a(P) - \varphi_a(Q), \varphi_{a,k}(P) - \varphi_{a,k}(Q)$.

We prepare sketches for all possible scales $R = c^i$, $i \in [\log_c d]$, for c a sufficiently large constant, determined later. For each scale we solve the threshold problem: output "far" if $\mathrm{ed}(P,Q) \geq R$ and output "close" if $\mathrm{ed}(P,Q) \leq R/c$, with probability at least $2/3$ (this can be amplified to whp by taking independent sketches). We also assume that $\mathrm{ed}(P,Q) \leq cR$ since the algorithm can enumerate all scales R from the biggest to the smallest stopping as soon as the sketch for the corresponding scale outputs "far".

The main idea is the following. Call $a \in \Sigma$ *expensive character* if $a \in T_{1/2}$, where T_δ is the set of characters z such that $\|\zeta_{z,k}\|_1 > \delta$ from some $k \in K$. In other words, the expensive characters are the ones that contribute a constant fraction to the edit distance (through ζ_a's). To find the expensive characters, we down-sample the characters to some set S such that there are few expensive characters in S. It remains to estimate the number of expensive characters in S.

For an expensive character a, we say it is *expensive at scale k* for some $k \in K$ if $\|\zeta_{a,k}\|_1 > 1/2$. The main observation is that if a is an expensive character at scale k, then $\|\zeta_{a,k}\|_1 \geq \frac{1}{\text{polylog}(d)}\|\{\zeta_{a,k}\}_{a\in S}\|_1$ (this step uses the second part of Theorem 5.2.1). Now, to find such characters a, we use a sketching algorithm for finding *heavy hitters* under ℓ_1. A more detailed description follows.

Definition 6.2.3. *Consider a vector $v \in \mathbb{R}^m$. Coordinate i is called a ϕ-*heavy hitter *if $|v_i| \geq \phi\|v\|_1$. The set of ϕ-heavy hitters is denoted by $\text{HH}_1(\phi)$.*

We will use the CountMin algorithm due to Cormode and Muthukrishnan [CM05a].

Lemma 6.2.4 ([CM05a])**.** *There exists a (randomized) sketching algorithm $\mathbf{sk}(\cdot)$, that for every two input vectors $x, y \in \mathbb{R}^m$ and parameter $\phi > 0$, computes sketches $\mathbf{sk}(x)$ and $\mathbf{sk}(y)$ of size $O(\frac{1}{\phi}\log m)$, such that one can reconstruct from the two sketches a set $H \subset [d]$, where, denoting $\text{HH}_1(\phi)$ the heavy hitters for the vector $x - y$, we have*

$$\Pr[\text{HH}_1(\phi) \subseteq H \subseteq \text{HH}_1(\tfrac{\phi}{2})] \geq 1 - m^{-\Omega(1)}.$$

The algorithm. Fix some R. Remember that we are trying to decide whether $E \geq R$ or $E \leq R/c$, where $E = \underline{\text{ed}}(P, Q)$ (and by assumption $E \leq cR$). Let $S \subset [d]$ be a multi-set of size $|S| = \frac{d}{R} \cdot c_S \log d$, chosen uniformly at random with replacement, where c_S is a large constant.

We want to find, for each $k \in K$, the expensive characters $a \in S$ at scale k. Fix $k \in K$ and consider the multi-set of vectors $\Upsilon_k(P) = \{2k\varphi_{a,k}(P)\}_{a\in S}$, and thus $\Upsilon_k(P) \subseteq \{0,1\}^d$. For $l = \frac{d}{2k} \cdot c_L \log d$, we sub-sample a multi-set $L \subset [d]$ of l coordinates (with replacement). For each $\varphi_{a,k} \in \Upsilon_k(P)$, we let $\varphi^L_{a,k}$ be the restriction of $\varphi_{a,k}$ to the coordinates from L, and let $\Upsilon^L_k(P)$ be the set of $\varphi^L_{a,k}$, where $\varphi_{a,k}$ ranges over $\Upsilon_k(P)$.

Our actual sketch consists of applying the CountMin algorithm (Lemma 6.2.4) to each set $\Upsilon^L_k(P)$, $k \in K$, to find the positions of all the non-zeros in all the vectors in $\Upsilon^L_k(P)$. For this we view $\Upsilon^L_k(P)$ as a vector of size $|S| \cdot l$, and apply CountMin with $\phi = \Omega(1/\log^4 d)$.

The reconstruction stage for some sketches $\mathbf{sk}(P)$ and $\mathbf{sk}(Q)$ is then the following. Define $\Upsilon_k = \Upsilon_k(P) - \Upsilon_k(Q) = \{2k\zeta_{a,k}(P)\}_{a\in S}$, and similarly with Υ^L_k. Using the linear in φ CountMin sketch $\mathbf{sk}(P)-\mathbf{sk}(Q)$, we find all the non-zeros in Υ^L_k. Once we found all the non-zeros in Υ^L_k, we define the set $X_k \subset S$ of characters which are "near-expensive" according to the subset L: i.e., $a \in X_k$ iff $\|2k\zeta^L_{a,k}\|_1 \geq \frac{1}{3}c_L \log d$. Then, the set of all expensive characters in S is estimated to be $X = \cup_k X_k$. If $|X| \geq \frac{1}{3} \cdot c_S \log d$, then we declare P and Q to be far. Otherwise, P and Q are close.

Analysis. We now show the correctness of the algorithm.

Let T^S_δ be the set of characters $a \in S$ such that there exists $k \in K$ for which $\|\zeta_{a,k}\|_1 \geq \delta$. The main part of the analysis is to prove that the set X satisfies the following with probability $\geq 2/3$:

$$T^S_{1/2} \subseteq X \subseteq T^S_{1/5}. \tag{6.3}$$

Eqn. (6.3) will be enough due to the following estimates of the sizes of $T^S_{1/2}$ and $T^S_{1/5}$. We have that $\mathbb{E}_S\left[|T^S_\delta|\right] = |T_\delta|/d \cdot |S| = \frac{|T_\delta|}{R} \cdot c_S \log d$, and, by Lemma 5.2.3, $1/2 \cdot E \leq |T_\delta| \leq 4/\delta \cdot E$. Then, if P and Q are far, then $\mathbb{E}_S\left[|T^S_{1/2}|\right] \geq \frac{1}{2} \cdot c_S \log d$, and then, by Chernoff bound, $|T^S_{1/2}| \geq \frac{1}{3} \cdot c_S \log d$, w.h.p. Similarly, if P and Q are close, then $\mathbb{E}_S\left[|T^S_{1/5}|\right] \leq \frac{20}{c} \cdot c_S \log d$, and thus, by Chernoff bound, $|T^S_{1/5}| \leq \frac{30}{c} \cdot c_S \log d$, w.h.p. Setting $c > 30 \cdot 3 = 90$ will separate the "far" from the "close" case.

Next we argue that restricting to coordinates from L roughly preserves the property of being expensive character at scale k. First, if $a \in S$ was expensive character at scale k, i.e., $\|\zeta_{a,k}\| > 1/2$, then we argue that w.h.p. $\|2k\zeta^L_{a,k}\|_1 > \frac{1}{3}c_L \log d$. Indeed $\mathbb{E}_L\left[\|2k\zeta^L_{a,k}\|_1\right] = \frac{\|2k\zeta_{a,k}\|_1}{d} l = \frac{1}{2} \cdot c_L \log d$, and thus, w.h.p., $\|2k\zeta^L_{a,k}\|_1 \geq \frac{1}{3} \cdot c_L \log d$ (by Chernoff bound). Second, in a similar fashion we get that, if $\|\zeta_{a,k}\| < 1/5$, then $\|2k\zeta^L_{a,k}\| < \frac{1}{4} \cdot c_L \log d$, w.h.p.

Now we bound the number of non-zeros in the vector Υ^L_k — this would set a lower bound for the value ϕ, and thus an upper bound on the space. For this, we just need to estimate $\mathbb{E}_{S,L}\left[\|\Upsilon^L_k\|_1\right]$:

$$\mathbb{E}_{S,L}\left[\|\Upsilon^L_k\|_1\right] = |S| \cdot l \cdot \frac{1}{d^2} \sum_{a\in[d], i\in[d]} |2k\zeta_{a,k,i}| \leq \frac{c_S c_L \log^2 d}{R} \cdot \|\zeta\|_1 \leq \frac{c_S c_L \log^2 d}{R} \cdot O(\log^2 d) \cdot E \leq O(\log^4 d),$$

where we used the second part of Theorem 5.2.1 to upper bound $\|\zeta\|_1$. By Markov's inequality, $\mathbb{E}_{S,L}\left[\|\Upsilon^L_k\|_1\right] \leq O(\log^4 d)$ with probability ≥ 0.9.

At this moment we are done. The set X will contain all the characters a such that there exists k for which $\|\zeta_{a,k}\| > 1/2$, or $a \in T^S_{1/2}$. Also, for any $a \in X$, there exists some k such that $\|\zeta_{a,k}\| > 1/5$, which means that $X \subseteq T^S_{1/5}$. Thus, Eqn. (6.3) holds with probability $\geq 8/10$, which is what we wanted to prove for correctness.

The space requirement is $O(\log d \cdot 1/\phi \cdot \log d) = O(\log^6 d)$. □

6.2.2 Implementing the sketch in the streaming model

We now show how to modify our sketching algorithm from Section 6.2 in order to make it computable in a streaming fashion. Specifically, we show we can compute the sketch having only a sequential access to the permutation $P \in \text{Ulam}_d$ in order from $P[1], P[2], \ldots$ to $P[d]$, and we are bounded to use only $(\log d)^{O(1)}$ space.

We use an algorithm for block heavy hitters, due to [ADI08], described below. Then we show how to prove Theorem 6.2.1.

Block Heavy Hitters

Let $M_{n,m}$ be the set of real matrices A of size n by m, with entries from $E = \frac{1}{nm} \cdot \{0, 1, \ldots nm\}$. For a matrix A, we let A_i be its i^{th} row.

Theorem 6.2.5 ([ADI08]). *Fix some $n, m \geq 1$ and $\phi \in [0, 1]$. There exists a randomized linear map (sketch) $\mu : M_{n,m} \to \{0,1\}^s$, where $s = (\phi \log nm)^{O(1)}$, such that the following holds. For a matrix $A \in M_{n,m}$, it is possible, given $\mu(A)$, to find a set $W \subset [n]$ of rows such that, with high probability, we have:*

- *for any $i \in W$, $\frac{\|A_i\|_1}{\|A\|_1} \geq \phi/2$ and*
- *if $\frac{\|A_i\|_1}{\|A\|_1} \geq \phi$, then $i \in W$.*

Also, there exists a randomized function $\rho : E^m \to \{0,1\}^s$ such that $\mu(A) = \mu'(\rho(A_1), \rho(A_2), \ldots \rho(A_n))$ (i.e., the sketch μ may be seen as first sketching the rows of A (using the same function ρ) and then sketching the sketches).

For completeness purposes, we offer a description of the algorithm for computing the sketch μ. The algorithm of [ADI08] is somewhat similar to the CountMin sketch of [CM05a] and is as follows. The function $\rho(x)$ for $x \in E^m$ is a vector of $C = (\log nm)^{O(1)}$ Cauchy projections of x. In particular, $[\rho(x)]_i$, for $i \in [C]$, is equal to $[\rho(x)]_i = v_i \cdot x$, where each entry of $v_i \in \mathbb{R}^m$ is chosen according to the Cauchy distribution with pdf $f(z) = \frac{2}{\pi} \frac{1}{1+z^2}$. Then μ hashes $\rho(A_1), \rho(A_2), \dots \rho(A_n)$ (with keys $1, 2, \dots n$ respectively) to a set of $H = (\log nm)^{O(1)}$ hash tables, each of size $L = (\phi^{-1} \log nm)^{O(1)}$. We note that an entry of an hash table is now a vector of size C that is equal to the sum of all vectors $\rho(A_i)$ that hash into the corresponding bucket. The reconstruction stage finds all elements $W \subset [n]$ that are "heavy" in at least a constant fraction of the H hash tables, where we say an element $i \in [n]$ is "heavy" in a hash table in the following case. Consider the corresponding hash table, and the bucket where i is hashed to. If the median of the absolute value of the C values stored in the corresponding bucket is greater than some $\frac{\phi^{O(1)}}{(\log nm)^{O(1)}}$, then i is "heavy" in the corresponding hash table.

A Streamable Sketch for Ulam metric: proof of Theorem 6.2.1

Fix $P \in \text{Ulam}_d$. We use the same notation as in Section 5.2.1 and Theorem 6.2.2.

Algorithm. We first slightly modify our embedding ϕ, to obtain another constant distortion embedding $\psi : \text{Ulam}_d \to \bigoplus_{(\ell_2)^2}^{d} \bigoplus_{\ell_\infty}^{O(\log d)} \ell_1^{2d}$; the advantage of ψ is that it is more amenable to streaming. We define a vector $\psi_{a,k}(P)$ to be almost as $\varphi_{a,k}$, except that different entries are scaled. Namely for each $b \in \bar{\Sigma}$ such that $\varphi_{a,k} > 0$ (i.e., $b \in P_{ak}$), we set $\psi_{a,k} = \varphi_{a,k}(P) \cdot e^{-w/k} = \frac{1}{2k} e^{-w/k}$ where $w = P^{-1}[a] - P^{-1}[b]$ is the distance between the position of symbols a and b. As before, we construct $\psi = \oplus_{a \in [d]} \oplus_{k \in [K]} \psi_{a,k} \in \bigoplus_{(\ell_2)^2}^{d} \bigoplus_{\ell_\infty}^{O(\log d)} \ell_1^{2d}$. We note that this is constant distortion according to the following argument for some permutations $P, Q \in \text{Ulam}_d$. If we define T'_δ to be set of $a \in [d]$ such that there exists $k \in [K]$ such that $\|\psi_{a,k}(P) - \psi_{a,k}(Q)\|_1 \geq \delta$, then it is easy to note that $T_{1/2} \subset T'_{1/2e}$ and that, for any $\delta > 0$, we have $T'_\delta \subset T_{\delta/c}$ for some constant $c > 1$. The last claim follows from the argument that, for any $a \in [d] \setminus T_{\delta/c}$, we have that

$$\|\psi_{a,k}(P) - \psi_{a,k}(Q)\|_1 \leq \sum_{j=1}^{d/k} e^{-j+1} \|\psi_{a,kj}(P) - \psi_{a,kj}(Q)\|_1 \leq \sum_{j=1}^{d/k} e^{-j+1} \cdot \delta/c \leq \delta.$$

The streamable sketching then proceeds as follows. As before, we subsample a set S of coordinates of size $|S| = \frac{d}{R} \cdot c_S \log d$. As in Theorem 6.2.2, we construct a separate sketch for each $k \in [K]$. So fix k. Consider the matrix A of size $n = |S|$ and $m = |\bar{\Sigma}| = 2d$ to be equal to $A^{P,k}_{a,b} = [\psi_{a,k}]_b$ (the n^{th} coordinate of $\psi_{a,k}$. We will apply the linear sketch of [ADI08]. Let μ, ρ be as in the Theorem 6.2.5.

During the streaming operation, we keep a sketch $\rho(A^{P,k}_a)$, where $a \in [d]$ is the symbol in the current position in the stream. Once we move the next position, $P^{-1}[a] + 1$, which contains a symbol $b = P[P^{-1}[a] + 1]$, we perform the following two steps. First, we multiply $\rho(A^{P,k}_a)$ by $e^{-1/k}$, to obtain $e^{-1/k} \rho(A^{P,k}_a) = \rho(A^{P,k}_b)$. Second, we add the sketch $\rho(A_b)$ to the (linear) sketch μ.

In the end, our sketch $\mathbf{sk}(P)$ consists of k sketches μ, for each thereshold radius R.

Analysis. Fix permutations $P, q \in \text{Ulam}_d$. The analysis is now quite very simple: we just need to verify that, for fixed $k \in [K]$, $\mu(A^{P,k})$ and $\mu(A^{Q,k})$ lets us identify all characters a that are expensive at scale k. Indeed $\mu(A^{P,k}) - \mu(A^{Q,k}) = \mu(A^{P,k} - A^{Q,k})$ and thus, the algorithm from Theorem 6.2.5 will return a set W_k such that:

- for any $a \in W_k$, $\frac{\|\psi_{a,k}(P)-\psi_{a,k}(Q)\|_1}{\sum_{a\in S}\|\psi_{a,k}(P)-\psi_{a,k}(Q)\|_1} \geq \phi/2$ and
- if $\frac{\|\psi_{a,k}(P)-\psi_{a,k}(Q)\|_1}{\sum_{a\in S}\|\psi_{a,k}(P)-\psi_{a,k}(Q)\|_1} \geq \phi$, then $i \in W_k$.

Remember that we use $\phi = (\log d)^{-O(1)}$. Now, combined with the observation that

$$\mathbb{E}_S\left[\sum_{a\in S}\|\psi_{a,k}(P)-\psi_{a,k}(Q)\|_1\right] \leq \mathbb{E}_S\left[\sum_{a\in S}\|\varphi_{a,k}(P)-\varphi_{a,k}(Q)\|_1\right] \leq O(\log^2 d),$$

we conclude that the set $W = \cup_k W_k$ is such that ${T'}^S_{1/20} \subseteq W \subseteq {T'}^S_{1/5}$, and thus by estimating $|W|$, we can decide whether $\mathrm{ed}(P,Q) \leq R$ or $\mathrm{ed}(P,Q) \geq cR$ w.h.p.

This completes the proof of Theorem 6.2.1.

Part II

Impossibility Results

Chapter 7

Sketching Lower Bounds

In this chapter, we will show that the classical approaches to NN under edit, Ulam, and EMD distances fail to achieve good approximation.

For edit and Ulam distances, we show a sketching lower bound for these two distances, for up to a nearly logarithmic approximation. An immediate implication is that any embedding of edit or Ulam distances into ℓ_1, or powers of ℓ_2, must have at least a near-logarithmic distortion. This is the *first* non-trivial sketching lower bound for any edit distance.

For EMD over boolean cube $\{0,1\}^d$, we show a sketching lower bound for up to linear in the dimension approximation. Again, this implies non-embeddability statements into, say, ℓ_1 or powers of ℓ_2 spaces.

Edit and Ulam distances. For edit distances, our result should also be seen from a wider perspective — that of understanding the fundamental complexity of the edit distances. In particular, currently, we lack efficient algorithms for most problems involving standard edit distance (even on binary strings). For the NN problem, all known algorithms with sublinear query time either require large space or have large approximation error. Specifically, Indyk [Ind04] achieves constant approximation using $n^{d^{\Omega(1)}}$ space, and Ostrovsky and Rabani [OR07] obtain $2^{O(\sqrt{\log d \log\log d})}$ approximation using space that is polynomial in d and n. Similarly, even the basic question of computing the edit distance between two strings requires nearly-quadratic time, and the best approximation achievable in a near-linear time is of $2^{O(\sqrt{\log d \log\log d})}$ (see also the discussion in Chapter 6).

It is thus natural to ask: is it really "hard" to design algorithms for the edit distance? A natural benchmark is the Hamming distance, which is equal to the number of positions where the two strings differ. Hamming distance can be seen as edit distance where the only operations allowed are substitution. For Hamming distance, much better algorithms are known: (i) the distance between two strings can clearly be computed in $O(d)$ time, and (ii) NN schemes of [KOR00, IM98] achieve $1+\epsilon$ approximation using space that is polynomial in d and in n^{1/ϵ^2} (see Sections 3.1 and 3.5.1). Empirically, edit distance appears to be more difficult than Hamming distance, and the reason is quite clear — insertions and deletions cause portions of the string to move and create an alignment problem — but there is no rigorous evidence that supports this intuition. In particular, we are not aware of a computational model in which the complexity of approximating edit distance is provably larger than that of Hamming distance. (See also Section 7.7 for a related discussion.)

The result from this chapter is the first rigorous evidence for the *computational* hardness of approximating the edit distance, to the best of our knowledge. In fact, we show that, in the sketching model, the complexity of estimating edit distance is significantly larger than that of Hamming distance, and this is the first setting where such a separation is known.

Furthermore, from a technical perspective, we note that, prior to this thesis, the only sketching lower bounds were known for the ℓ_p spaces, for $p \in [1, \infty]$ [Woo04, SS02a, BJKS04]. Only very recently, [JW09] have managed to also prove a sketching lower bound for some product spaces (called mixed-ℓ_p/cascaded norms in their paper). Sketching of edit distance was studied in [BEK+03, BJKK04, OR07, CK06], but the only known sketching lower bound was trivial in the sense that it followed immediately from the Hamming distance and was uninformative for even a 2-approximation.

Statement of results. We now formally state our sketching lower bound for the edit (on $\{0,1\}^d$) and Ulam distances. Our result is best expressed in terms of the communication complexity of the *distance threshold estimation problem (DTEP)* [SS02a]. In DTEP, for a threshold R and an approximation $\alpha \geq 1$ fixed as parameters, we are given inputs x, y and we want to decide whether $\mathrm{ed}(x, y) > R$ or $\mathrm{ed}(x, y) \leq R/\alpha$.

In the *communication protocols setting*, Alice and Bob, who have access to a common source of randomness, receive strings x and y respectively as their inputs, and their goal is to solve DTEP by exchanging messages. The communication complexity of the protocol is then defined as the minimum number of bits Alice and Bob need to exchange in order to succeed with probability at least 2/3. When x, y come from the standard edit metric, we denote the communication complexity by $\mathrm{CC}_{\alpha,R}^{\{0,1\}^d}$. Similarly, when x, y come from the Ulam metric, we denote the communication complexity by $\mathrm{CC}_{\alpha,R}^{\mathrm{Ulam}_d}$. Here we prove the following theorem on the communication complexity of DTEP for edit and Ulam distances, exhibiting a trade-off between communication and approximation.

Theorem 7.0.6. *There exists a constant $c > 0$, such that for every string length $d > 1$, approximation $\alpha > 1$, and R satisfying $d^{0.1} \leq R \leq d^{0.49}$, we have*

$$\mathrm{CC}_{\alpha,R}^{\mathrm{Ulam}_d} \geq c \log\left(\tfrac{\log d}{\alpha \log \alpha}\right).$$

In particular, in the regime of constant communication complexity $\mathrm{CC}_{\alpha,R}^{\mathrm{Ulam}_d} = O(1)$, the approximation must satisfy $\alpha = \Omega\left(\frac{\log d}{\log\log d}\right)$.

The same holds for the edit distance on binary strings as well: $\mathrm{CC}_{\alpha,R}^{\{0,1\}^d} \geq c \log\left(\frac{\log d}{\alpha \log \alpha}\right)$.

We note that the communication complexity of the DTEP problem for a metric is a lower bound on the sketching complexity of the metric (for the same approximation). This follows from the fact that a sketching algorithm is a particular type of a communication protocol: Alice computes the sketch of x and sends it to Bob, who, using the sketch of x and the sketch of y, solves the DTEP problem.

To compare our lower bound with the Hamming distance, note that the Hamming distance admits a sketching algorithm achieving $1+\epsilon$ approximation in $O(1/\epsilon^2)$ space [KOR00]. In particular, for a 2-approximation, the complexity of the Hamming metric is $O(1)$, while that of the edit distance is at least $\Omega(\log\log d)$, by the above theorem. It thus follows that the edit distance is indeed provably harder to compute than the Hamming distance, at least in the context of communication protocols.

The previously known lower bounds for $\mathrm{CC}_{\alpha,R}^{\{0,1\}^d}$ and $\mathrm{CC}_{\alpha,R}^{\mathrm{Ulam}_d}$ are all obtained by a straightforward reduction from the same problem on Hamming metric. These bounds assert that the communication complexity for $\alpha = 1+\epsilon$ is $\Omega(1/\epsilon)$, and in the case of sketching protocols $\Omega(1/\epsilon^2)$ [Woo04] (see also [Woo07, Chapter 4] and [BC09]), and both are clearly uninformative for (say) $\alpha = 2$. See also [SU04] for other related results. The only non-trivial upper bounds currently known are: (i) $\mathrm{CC}_{\alpha,R}^{\{0,1\}^d} \le O(1)$ for suitable $\alpha = 2^{O(\sqrt{\log d \log\log d})}$; and (ii) $\mathrm{CC}_{\alpha,R}^{\mathrm{Ulam}_d} \le O(1)$ for suitable $\alpha = O(\log d)$; and they both follow via embedding into ℓ_1 [OR07, CK06]. (See also Chapter 1, namely Section 1.3.2 and Table 1.3.)

Other implications. We note our theorem also implies non-embeddability results into ℓ_1 and powers of ℓ_2, as we show in the next corollary. (In Section 7.6, we also present a more direct proof, via a Poincaré-type inequality established in Theorem 7.0.6.)

Corollary 7.0.7. *For every fixed $p \ge 1$, embedding the standard edit metric or the Ulam metric into $(\ell_2)^p$, the p-th power of ℓ_2, requires distortion $\Omega\left(\frac{\log d}{\log\log d}\right)$. The same is true also for embedding into ℓ_1.*

We note that our lower bounds are near-tight in at least one case, namely that of embedding of Ulam distance into ℓ_1, which has $O(\log d)$ distortion embedding into ℓ_1 [CK06]. The only previous distortion lower bound was 4/3 [Cor03].

Proof of Corollary 7.0.7. Suppose $p \ge 1$ is fixed and the edit metric ed (or similarly the Ulam metric) embeds into $(\ell_2)^p$ with distortion $D \ge 1$. In other words, the metric $\mathrm{ed}^{1/p}$ (i.e. $1/p$-power of every distance) embeds into ℓ_2 with distortion $D^{1/p}$. Since ℓ_2 embeds arbitrarily well into ℓ_1, it is enough to solve DTEP for ℓ_1. For the DTEP problem on ℓ_1, [KOR00] give a sketching algorithm with approximation $1+\frac{1}{p}$ and communication $O(p^2)$. Together, we obtain a protocol for the DTEP problem on the metric $\mathrm{ed}^{1/p}$, which achieves approximation $D^{1/p}(1+\frac{1}{p})$ and communication $O(p^2)$. Observe that the same protocol solves also DTEP on the edit metric ed, except that the threshold now is R^p instead of R, and the approximation is $(D^{1/p}(1+\frac{1}{p}))^p < De$. The communication is the same $O(p^2)$, and thus Theorem 7.0.6 implies that $De\log(De) \ge 2^{-O(p^2)}\log d$. For fixed p this completes the proof. $\square$

Recent work. Finally, we remark that in very recent work [AJP10], we have extended the lower bounds from Theorem 7.0.6 to prove improved lower bounds on the communication complexity:

Fact 7.0.8 ([AJP10]). *There exists a constant $c > 0$, such that for every string length $d > 1$, approximation $\alpha > 1$, and R satisfying $d^{0.1} \le R \le d^{0.49}$, we have*

$$\mathrm{CC}_{\alpha,R}^{\mathrm{Ulam}_d} \ge c\frac{\log d/\log\log d}{\alpha}.$$

The same holds for the edit distance on binary strings as well: $\mathrm{CC}_{\alpha,R}^{\{0,1\}^d} \ge c\frac{\log d/\log\log d}{\alpha}$.

In particular, the fact implies that, for constant approximation, one requires $\Omega\left(\frac{\log d}{\log\log d}\right)$ communication. (In the regime of constant communication complexity, the above fact does not provide any improvement.) The lower bound from this fact is tight for the Ulam distance, up to the exponent of the logarithm, since we have shown an sketching protocol achieving constant approximation to Ulam distance in $\log^{O(1)} d$ space (see Chapter 6). The proof of Fact 7.0.8 from [AJP10] relies crucially on Theorem 7.0.6 proven in this thesis.

EMD. We also consider the EMD metric on the hypercube $\{0,1\}^d$, for which we prove the following communication complexity lower bound.

Theorem 7.0.9. *For every dimension $d \geq 1$, and approximation ratio $1 \leq \alpha \leq d$, there exists R such that the corresponding problem DTEP problem for EMD has communication complexity at least $\Omega(d/\alpha)$.*

We note that, in the regime of protocols with constant communication complexity, our lower bound says that the best achievable approximation is at least $\Omega(d)$. This is in fact tight, as there exists a $O(d)$ distortion embedding of EMD over $\{0,1\}^d$ into ℓ_1 (and thus a constant communication protocol by the sketch of [KOR00]), see, [Cha02, IT03, KN06].

Our result also implies that EMD over hypercube has $\Omega(d)$ distortion for embedding into ℓ_1, powers of ℓ_2, or ultra-sketchable spaces. Only non-embeddability into ℓ_1 was previously known [KN06].

The results from this chapter have previously appeared in [AK07, AJP10].

7.1 Our Techniques

We now describe the main techniques of the proof of Theorem 7.0.6 (Ulam and edit distance lower bound), and Theorem 7.0.9 (EMD lower bound). The two proofs share two general commonalities. The first one is a certain reduction from the question on general communication protocols to the question on very specific protocols with only *one bit* of communication. The second (more implicit) commonality is the use of Fourier decomposition of functions on the hypercube (see Section 7.1.2 below for a primer in Fourier analysis on hypercube and $\mathbb{Z}_p = \{0, 1, 2, \dots p-1\}$).

Common technique. Generally speaking, we design two input distributions: $\tilde{\mu}_0$ over "far" pairs (x, y) (i.e. $\mathrm{ed}(x, y) > R$), and $\tilde{\mu}_1$ over "close" pairs (i.e. $\mathrm{ed}(x, y) \leq R/\alpha$). The goal then becomes to show that these distributions are indistinguishable by protocols with low communication complexity. By Yao's minimax principle, it suffices to consider deterministic protocols.

The first step for both theorems is to reduce the problem to proving that the two distributions $\tilde{\mu}_0, \tilde{\mu}_1$ are indistinguishable by boolean functions. Roughly speaking, we show that if there is a protocol using at most l bits of communication, then there exists a (deterministic) sketching protocol that uses sketches of 1 bit and achieves an advantage of at least $\Omega(2^{-l})$ in distinguishing between the two distributions. Let $\mathcal{H}^A, \mathcal{H}^B$ be the boolean functions that Alice and Bob, respectively, use as their sketch functions. We can then further restrict the sketching protocol so that protocol accepts if and only if $\mathcal{H}^A(x) = \mathcal{H}^B(y)$. This step is shown in Section 7.2.

From this moment, the two theorems diverge.

EMD. For EMD, the rest of the proof is based on one basic idea. We would like to design the hard distributions $\tilde{\mu}_0$ and $\tilde{\mu}_1$ that are hard to distinguish by boolean functions $\mathcal{H}^A, \mathcal{H}^B$. The basic idea is to consider two extreme cases. First, if $\mathcal{H}^A, \mathcal{H}^B$ depend on many coordinates of the input, then they fail to distinguish the two distributions because of the random noise that we include in our hard distribution. Second, if $\mathcal{H}^A, \mathcal{H}^B$ depend on few coordinates, then again they fail to distinguish because of a specific property of our hard distribution for EMD. This property on the hard distribution induces certain Fourier-analytic structure on the functions $\mathcal{H}^A, \mathcal{H}^B$, namely that all low-level Fourier coefficients have to be zero. Our hard distribution satisfying the above two properties (random noise, and the Fourier-analytic property) is similar to the one from [KN06] used for the ℓ_1 non-embeddability of EMD. Finally, to conclude, we argue that considering the two cases — $\mathcal{H}^A, \mathcal{H}^B$ depend on either many or few coordinates — is enough due to the Fourier decomposition of the functions. The proof of Theorem 7.0.9 appears in Section 7.2.

Ulam and edit distances. For Ulam and edit distances, the theorem requires considerably more ideas. On a basic level, we reuse the idea of decomposing the functions into parts that depend on either a few or many coordinates. While the case of "depending on many coordinates" is essentially the same, the other case is different (from either the EMD case of the edit distance non-embeddability results) and is the main thrust of the sketching lower bound for Ulam and edit distances (as well as of the majority of the rest of this chapter).

The second step of this lower bound will be to further characterize the advantage achieved by $\mathcal{H}^A, \mathcal{H}^B$ in terms of a carefully crafted measure of statistical distance between the two input distributions $\tilde{\mu}_0, \tilde{\mu}_1$. For this approach to be effective, it is important that the functions $\mathcal{H}^A, \mathcal{H}^B$ depend only on a few coordinates of their inputs, and in order to guarantee this (indirectly), we include in $\tilde{\mu}_0, \tilde{\mu}_1$ a noise component, which effectively destroys any dependence of $\mathcal{H}^A, \mathcal{H}^B$ on many coordinates. Specifically, this step assumes that, each distribution $\tilde{\mu}_t$, $t \in \{0, 1\}$, has the following structure: choose $x \in \mathbb{Z}_p^d$ uniformly at random, and then generate y from x via a sequence of two randomized operations. The first of the two is a noise operator with rate $\rho \in (0, 1)$, i.e., each coordinate is modified independently with probability $1 - \rho$ into a randomly chosen value. The second operation permutes the coordinates according to a permutation drawn from a distribution $\mathcal{D}_t$. Given this $\mathcal{D}_t$, consider the following derived distribution: take a vector $u \in \mathbb{Z}_p^d$ with λ non-zero positions (called a λ-*test*) and apply a random permutation $\pi \leftarrow \mathcal{D}_t$ to it; let $A_u^{(t,\lambda)}$ be the resulting distribution of vectors. (Note that the support of $A_u^{(t,\lambda)}$ contains only vectors with precisely λ non-zero entries.) Our measure Δ_λ, *called λ-test distinguishability*, is the maximum, over all such λ-tests u, of the total variation distance between $A_u^{(0,\lambda)}$ and $A_u^{(1,\lambda)}$. It pretty much captures the statistical advantage in distinguishing $\mathcal{D}_0$ from $\mathcal{D}_1$ (and thus $\tilde{\mu}_0$ from $\tilde{\mu}_1$) achievable by inspecting only λ positions of, say, y (e.g., by tracing them back to x). Altogether, our upper bound on the advantage achieved by $\mathcal{H}^A, \mathcal{H}^B$ takes roots in the following dichotomy. If $\mathcal{H}^B$ essentially depends on many coordinates of y (e.g., a linear function with many terms), then the advantage is bounded by ρ^λ (i.e., the noise destroys almost all the information), and if $\mathcal{H}^B$ essentially depends on a few, say λ, coordinates, then the advantage is bounded by the aforementioned Δ_λ. To prove this dichotomy, we rely on Fourier analysis which expands $\mathcal{H}^A, \mathcal{H}^B$ into linear functions at different levels λ. This step appears in Section 7.4.

We finalize by completing the description of $\tilde{\mu}_0, \tilde{\mu}_1$, namely by detailing the construction of $\mathcal{D}_0, \mathcal{D}_1$. For the construction distributions, we give an upper bound on the λ-test distinguishability Δ_λ for these distributions. In a simplified view, each distribution $\mathcal{D}_t$ is generated by a block rotation operation, namely, choosing a random block of length L and applying to it $\epsilon_t L$ cyclic shifts. The difference between the two distributions is in the magnitude of the rotation (namely, ϵ_t). This step appears in Section 7.5.

Our use of Fourier analysis is elementary, and does not involve the KKL theorem [KKL88] or Bourgain's noise sensitivity theorem [Bou02], which were used in the previous non-embeddability results for edit distance [KN06, KR06]. We also note that our hard distribution is notably different from the distributions of [KR06] or [KN06], which do admit efficient communication protocols.

7.1.1 Additional notation

Before proceeding to the proof, we establish some additional notation, and give a brief overview of the Fourier analysis, as used in the proof of our theorem.

For $x \in \Sigma^d$, we let x_i denote the i^{th} position in x whenever $i \in [d]$, and extend the notation to $i \notin [d]$ by defining $x_i = x_j$ where $i \equiv j \pmod d$ and $j \in [d]$.

We will use the following operation on strings.

Definition 7.1.1 (Rotation operations)**.** *Fix a positive integer d and an alphabet Σ. For $s, L \in [d]$, define the* right rotation operation $\overrightarrow{R}_{s,L} : \Sigma^d \to \Sigma^d$ *as follows. When applied to a string x, it takes the substring of x of length L starting at position s (with wrap-around), and performs on it one cyclic shift to the right (by 1 position); the rest of x remains unchanged. A* left rotation $\overleftarrow{R}_{s,L}$ *is defined similarly. We call L the* length *of the rotation operation.*

See an illustration in Fig. 7-1.

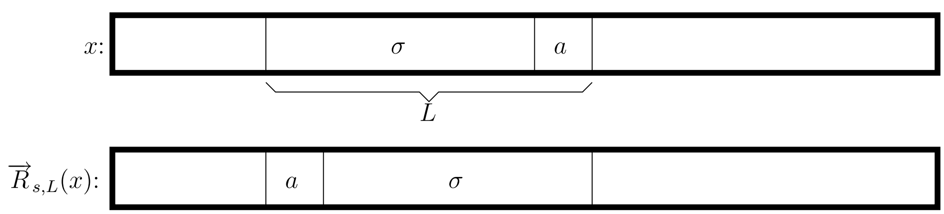


Figure 7-1: The rotation operation. Here, σ is the substring of length $L-1$ starting at position s in x, and a is the character at position $s+L-1$ in x.

Note that $\overrightarrow{R}_{s,L}$ works as a permutation (and thus is a bijection on the space of strings). Also, for $i \in [L]$, $\left(\overrightarrow{R}_{s,L}\right)^i$ is a rotation of the same block by i positions to the right. Note that a rotation operation $\overrightarrow{R}_{s,L}$ can be simulated by at most two deletions and two insertions (and only one of each when the rotation block does not wrap-around at the string's boundary). Thus, $\mathrm{ed}\left(x, \left(\overrightarrow{R}_{s,L}\right)^i(x)\right) = O(i)$ for every x and i.

7.1.2 Fourier Analysis over $\mathbb{Z}_p^d$

We review basic Fourier Analysis over $\mathbb{Z}_p^d$ for a prime $p \geq 2$ (see, e.g., [Šte00] for more background on Fourier analysis, especially its use in theoretical Computer Science).

The collection of functions $f : \mathbb{Z}_p^d \to \mathbb{C}$ is a vector space of dimension p^d, equipped with an inner product given by $\langle f, g \rangle = \mathbb{E}_{x \in \mathbb{Z}_p^d}\left[f(x) \cdot \overline{g(x)}\right]$. For $u \in \mathbb{Z}_p^d$, define a character $\chi_u(x) = e^{\frac{2\pi i}{p}(x \cdot u)}$, where $x \cdot u$ is the scalar product of $x, u \in \mathbb{Z}_p^d$. The set of characters $\{\chi_u \mid u \in \mathbb{Z}_p^d\}$ forms an orthonormal basis, called the Fourier basis. Thus every function $f : \mathbb{Z}_p^d \to \mathbb{C}$ admits a Fourier expansion $f = \sum_{u \in \mathbb{Z}_p^d} \hat{f}_u \chi_u$, where $\hat{f}_u = \langle f, \chi_u \rangle$ is called the Fourier coefficient of f corresponding to u. Parseval's identity states that $\mathbb{E}_{x \in \mathbb{Z}_p^d}\left[f(x)\overline{g(x)}\right] = \sum_{u \in \mathbb{Z}_p^d} \hat{f}_u \overline{\hat{g}_u}$.

We let N_ρ stand for a *noise* vector over $\mathbb{Z}_p^d$, namely, a vector where each coordinate is set independently at random as follows: with probability ρ it is set to zero, and with probability $1-\rho$ it is set to a random value from $\mathbb{Z}_p$. We refer to ρ as the *rate* of the noise.

The *noise operator* T_ρ (also called Bonami–Beckner operator) operates on functions $f : \mathbb{Z}_p^d \to \mathbb{R}$, and is defined by $(T_\rho f)(x) = \mathbb{E}_{N_\rho}[f(x+N_\rho)]$. The following standard fact relates the Fourier coefficients of f with those of $T_\rho f$. For a vector $u \in \mathbb{Z}_p^d$, define the *weight* of u, denoted $\mathrm{wt}(u)$, to be the number of coordinates in u that are non-zero.

Fact 7.1.2. *For every vector $u \in \mathbb{Z}_p^d$, $\widehat{(T_\rho f)}_u = \hat{f}_u \cdot \rho^{\mathrm{wt}(u)}$.*

Proof. We can write $(T_\rho f)(x) = \mathbb{E}_{N_\rho}[f(x+N_\rho)]$ as

$$\mathbb{E}_{N_\rho}\left[\sum_{u\in\mathbb{Z}_p^d} \hat{f}_u e^{\frac{2\pi i}{p}u\cdot(x+N_\rho)}\right] = \sum_{u\in\mathbb{Z}_p^d} \hat{f}_u e^{\frac{2\pi i}{p}u\cdot x}\mathbb{E}_{N_\rho}\left[e^{\frac{2\pi i}{p}u\cdot N_\rho}\right] = \sum_{u\in\mathbb{Z}_p^d} \hat{f}_u \rho^{\mathrm{wt}(u)}\chi_u,$$

where we used the fact that for every $w \in \mathbb{Z}_p \setminus \{0\}$ we have $\mathbb{E}_{v\in\mathbb{Z}_p}\left[e^{\frac{2\pi i}{p}wv}\right] = 0$. □

Note that, for $p = 2$, i.e. Fourier expansion over $\{0,1\}^d$, this is equivalent to having $\widehat{(T_\rho f)}_S = \hat{f}_S \rho^{|S|}$ for every $S \subseteq [d]$.

7.2 Protocols with Constant Communication

We now develop a characterization of the constant-communication protocols for DTEP problem. Part of this characterization will be used in the rest of the chapter. Fix some metric $(\mathcal{M}, \mathsf{d}_\mathcal{M})$, threshold $R > 0$, and approximation $\alpha > 1$. As before, we call a pair (x,y) *far* if $\mathsf{d}_\mathcal{M}(x,y) > R$ and *close* if $\mathsf{d}_\mathcal{M}(x,y) \le R/\alpha$.

First, we show that, if there exists an efficient communication protocol for a DTEP problem, then, for every two distributions $\tilde{\mu}_0$ from $\tilde{\mu}_1$ over far and close pairs respectively (candidate "hard distribution), there exist some real functions (in fact, boolean) with a non-negligible advantage in distinguishing the distribution $\tilde{\mu}_0$ from $\tilde{\mu}_1$. In other words, for protocols with constant communication, for all distributions $\tilde{\mu}_0, \tilde{\mu}_1$, there exist functions $f, h : \mathcal{M} \to \{-1,+1\}$ such that

$$\Pr_{(x,y)\leftarrow\tilde{\mu}_0}[(f(x)-g(y))^2] - \Pr_{(x,y)\leftarrow\tilde{\mu}_1}[(f(x)-g(y))^2] \ge \Omega(1). \tag{7.1}$$

We also show that condition from inequality (7.1) is also *necessary* for protocols of constant communication. In particular, if the communication complexity of the considered DTEP problem is at least some absolute constant, then there exist some "hard distributions" $\tilde{\mu}_0$ and $\tilde{\mu}_1$, for which any real-valued function cannot distinguish between the two with good advantage, i.e., the inequality (7.1) does not hold.

We now quantify and prove the above two directions. We note that, we need only the first direction in the rest of the chapter, namely for proving Theorems 7.0.6 and 7.0.9.

Lemma 7.2.1. *Fix some metric $(\mathcal{M}, \mathsf{d}_\mathcal{M})$, threshold $R > 0$, and approximation $\alpha > 1$. Let $\tilde{\mu}_0$ and $\tilde{\mu}_1$ be some arbitrary distributions over far and close pairs, respectively. If the communication complexity of the DTEP problem is bounded by an integer l, then there exist* *boolean functions $\mathcal{H}^A, \mathcal{H}^B : \mathbb{Z}_p^d \to \{-1, +1\}$, such that*

$$\Pr_{(x,y)\leftarrow\tilde{\mu}_0}[\mathcal{H}^A(x) \neq \mathcal{H}^B(y)] - \Pr_{(x,y)\leftarrow\tilde{\mu}_1}[\mathcal{H}^A(x) \neq \mathcal{H}^B(y)] \geq \tfrac{1}{3} \cdot 2^{-l}. \tag{7.2}$$

Note that the above inequality is equivalent to inequality (7.1) for boolean functions $f = \mathcal{H}^A$ and $g = \mathcal{H}^B$.

Proof. The idea is to reduce the general communication protocol to a simultaneous (i.e. sketching) protocol where Alice and Bob each send a sketch of one bit only, and the referee performs an equality test on these two bits. Then, using Yao's minimax principle, we easily obtain two deterministic boolean functions $\mathcal{H}^A$ and $\mathcal{H}^B$, completing the proof.

To accomplish the reduction, consider an actual l-bit (randomized) protocol Π. We construct a one-bit sketching protocol as follows: Alice and Bob make a random guess of the entire transcript of an l-bit protocol using the public coins, uniform over the space of all 2^l protocols (the guess is independent of the actual inputs). Each of them then checks whether the guessed transcript describes the messages they would send in the actual protocol Π, using the guessed transcript to simulate the other party's messages. For example, Alice starts the protocol Π (that depends on her input), but instead of sending the messages to Bob, she verifies that her messages are exactly the same as the ones appearing in the guessed protocol. Alice also uses the messages from the guessed protocol to simulate Bob's answers.

If at any moment Alice (or Bob) spots an inconsistency, she (or he) sends a bit chosen independently at random. Otherwise, Alice outputs 1, and Bob outputs the outcome of the guessed transcript. Observe that if the guessed transcript is not equal to the actual protocol they would have run, then at least one of the two players notices an inconsistency, and one of the bits output by Alice or Bob is random.

Thus, if x and y are such that $\mathrm{ed}(x, y) \leq R/\alpha$ (close pair), then Alice and Bob's bits are equal with probability at least $\frac{2}{3} \cdot 2^{-l} + (1 - 2^{-l})\frac{1}{2} = \frac{1}{2} + \frac{1}{6}2^{-l}$ (where $\frac{2}{3}$ is the probability that the original protocol Π succeeds on (x, y)). Similarly, if x and y are such that $\mathrm{ed}(x, y) > R$ (far pair), then Alice and Bob's bits are equal with probability at most $\frac{1}{3} \cdot 2^{-l} + (1 - 2^{-l}) \cdot \frac{1}{2} = \frac{1}{2} - \frac{1}{6}2^{-l}$. Using Yao's minimax principle, we conclude that, for given distributions $\tilde{\mu}_0$ and $\tilde{\mu}_1$ over far and close pairs respectively, there exist some fixed boolean functions $\mathcal{H}^A, \mathcal{H}^B$ that achieve a success probability at least $\frac{1}{2} + \frac{1}{6}2^{-l}$ on the distribution $\tilde{\mu} = \frac{\tilde{\mu}_0 + \tilde{\mu}_1}{2}$, or, formally,

$$\frac{1}{2} \Pr_{\tilde{\mu}_0}[\mathcal{H}^A(x) \neq \mathcal{H}^B(y)] + \frac{1}{2} \Pr_{\tilde{\mu}_1}[\mathcal{H}^A(x) = \mathcal{H}^B(y)] \geq \frac{1}{2} + \frac{1}{6} \cdot 2^{-l}.$$

We conclude that $\Pr_{\tilde{\mu}_0}[\mathcal{H}^A(x) \neq \mathcal{H}^B(y)] - \Pr_{\tilde{\mu}_1}[\mathcal{H}^A(x) \neq \mathcal{H}^B(y)] \geq \frac{1}{3} \cdot 2^{-l}$. ☐

We now show that, for the regime of constant communication, the above lemma is tight. Although the lemma from below will not be used in the rest of the chapter, we present it for the completeness purposes (we note that it has been used in [AJP10] to prove a certain direct-sum theorem on communication complexity).

Lemma 7.2.2. *Fix some metric $(\mathcal{M}, \mathsf{d}_{\mathcal{M}})$, threshold $R > 0$ and an approximation $\alpha > 1$, and assume that the corresponding DTEP problem has communication complexity equal to $C \geq 2$.*

Then, there exist some distributions $\tilde{\mu}_0$ and $\tilde{\mu}_1$ over far and close pairs respectively satisfying the following. For any vector-valued functions $f, g : \mathcal{M} \to \mathbb{R}^d$ with $\|f(x)\|_2, \|g(x)\|_2 \leq$

1 *for all* $x \in \mathcal{M}$ *and some dimension* $d \geq 1$*, there exists an* $\epsilon = O(C^{-1/4}\log^2 C)$ *such that*

$$\left|\mathbb{E}_{(x,y)\leftarrow\tilde{\mu}_1}\left[\|f(x)-g(y)\|^2\right] - \mathbb{E}_{(x,y)\leftarrow\tilde{\mu}_0}\left[\|f(x)-g(y)\|^2\right]\right| < \epsilon. \tag{7.3}$$

Note that, in Lemma 7.2.1, we obtain functions $\mathcal{H}^A, \mathcal{H}^B$ which may also be viewed as vector-valued functions, in a 1-dimensional space, of norm one.

Proof. Fix an ϵ such that $C = \Omega(1/\epsilon^4 \cdot \log^2 1/\epsilon)$. First we note that any protocol for the considered DTEP problem with success probability $\geq \frac{1}{2}+\epsilon/3$ has size at least $C' = \Omega(C\cdot\epsilon^2)$.

By Yao's principle, there exists a hard distribution ψ for protocols of size $< C'$. We decompose the distribution ψ into two distributions with distinct support. We define distribution $(x,y) \leftarrow \tilde{\mu}_0$ to be the distribution ψ conditioned on $\mathsf{d}_{\mathcal{M}}(x,y) > R$. Let p_0 be the probability that $\mathsf{d}_{\mathcal{M}}(x,y) > R$ when x, y are drawn from ψ. Analogously, define $\tilde{\mu}_1$ and p_1, and note that $p_1 = 1 - p_0$.

Now observe that $p_0, p_1 \geq \frac{1}{2} - \epsilon/3$ — otherwise, there exists a trivial 1-bit protocol with success probability at least $\frac{1}{2} + \epsilon/3$.

For the sake of contradiction assume Equation (7.3) does not hold, and, w.l.o.g.,

$$\mathbb{E}_{(x,y)\leftarrow\tilde{\mu}_1}\left[\|f(x)-g(y)\|^2\right] - \mathbb{E}_{(x,y)\leftarrow\tilde{\mu}_0}\left[\|f(x)-g(y)\|^2\right] \geq \epsilon.$$

Then, we show how to design a simultaneous-message protocol of size $O(1/\epsilon^2 \cdot \log^2 1/\epsilon) < C'$ that has success probability of at least $\frac{1}{2} + \epsilon/3$.

Namely, we take a randomized protocol that estimates the quantity $\|f(x)-g(y)\|^2$ up to additive $\epsilon/10$ term, with probability $1-\epsilon/10$, using the ℓ_2 estimation algorithm. Specifically, since $\|f(x)-g(y)\|^2 \leq 4$, we can just use a $(1+\epsilon/40)$-multiplicative ℓ_2 estimation protocol (e.g., via embedding ℓ_2 into the Hamming space and then using the [KOR00] sketch). Note that the protocol has size $O(1/\epsilon^2)$ (for [KOR00] sketch), times $O(\log 1/\epsilon)$ (to boost the success probability to $\geq 1-\epsilon/10$), times another $O(\log 1/\epsilon)$ (to guess the right scale); in other words, the size of the protocol is less than C'.

Let $z(x,y)$ be the estimate given by the ℓ_2 estimation protocol on input (x,y). Then the protocol accepts with probability exactly $z(x,y)$ (by tossing some additional random coins). The resulting success probability is at least:

$$\begin{aligned} &p_1 \cdot \mathbb{E}_{\tilde{\mu}_1}\left[(1-\epsilon/10)z(x,y)\right] + p_0 \cdot \mathbb{E}_{\tilde{\mu}_0}\left[(1-\epsilon/10)(1-z(x,y))\right] \geq \\ &\quad 1 - \epsilon/3 + \tfrac{1}{2}\left(\mathbb{E}_{\tilde{\mu}_1}\left[\|f(x)-g(y)\|^2\right] - \mathbb{E}_{\tilde{\mu}_0}\left[\|f(x)-g(y)\|^2\right]\right) - 3\epsilon/10 \geq \tfrac{1}{2} + \epsilon/3. \end{aligned}$$

This is a contradiction. Although the resulting protocol is randomized, by Yao's minimax principle, there should also exist a deterministic protocol with (at least) the same success probability since the input distribution is fixed.

This completes the proof of Lemma 7.2.2. □

7.3 Warm-up: EMD Lower Bound

Our first communication complexity lower bound is for the EMD distance over $\{0,1\}^d$, namely Theorem 7.0.9. This case should be seen as a warm-up for the considerably harder case of edit distances lower bound. Nonetheless, this case already uses the decomposition of boolean function into its Fourier decomposition, and some basic ides that will reappear in the theorem on the edit distances, presented in the following sections.

Proof of Theorem 7.0.9. We apply Lemma 7.2.1. Thus, to prove a lower bound of $\Omega(d/\alpha)$, we only need to construct hard distributions $\tilde{\mu}_0$ and $\tilde{\mu}_1$ over far and close pairs of sets (X, Y), respectively, such that, for any boolean functions $\mathcal{H}^A, \mathcal{H}^A$ over subsets of $\{0,1\}^d$, we have that

$$\Pr_{(X,Y)\leftarrow\tilde{\mu}_0}[\mathcal{H}^A(X) \neq \mathcal{H}^B(Y)] - \Pr_{(X,Y)\leftarrow\tilde{\mu}_1}[\mathcal{H}^A(X) \neq \mathcal{H}^B(Y)] \leq 2^{-\Omega(d/\alpha)}. \tag{7.4}$$

Specifically, from inequality (7.4) together with Lemma 7.2.1, we immediately conclude that the communication complexity is at least $\Omega(d/\alpha)$, proving Theorem 7.0.9.

To prove inequality (7.4), we build on the insights of the non-embeddability result of [KN06]. In particular, we use their code-based distribution of "hard inputs", as well as its Fourier-analytic properties.

We start by constructing the hard distributions $\tilde{\mu}_0$ and $\tilde{\mu}_1$. We assume that $1 < \alpha \leq d/200$, since for $\alpha > d/200$ the conclusion is trivial. Fix $R = d/100$. Fix $C \subset \{0,1\}^d$ to be a linear code with dimension $\geq d/4$ and weight $\geq cd$, where c is a constant; for existence of such code, see e.g. [KN06, Corollary 3.5]. For $x \in \{0,1\}^d$, we define Gx to be the set $\{x+a\}_{a\in C^\perp}$ (formally, G is the set of isometries $f_g(x) = x + g$, $g \in C^\perp$, and Gx is the orbit of x induced by the group action G). In the sequel, we will only consider as inputs (for Alice and Bob) sets of the form $X = Gx$ for some $x \in \{0,1\}^d$. Notice these sets all have size $s = |C^\perp| \leq 2^{3d/4}$. Furthermore, for all $X = Gx, Y = Gy$, and $y' \in Y$, we have $\mathrm{EMD}(X,Y) = \min_{x'\in X} \|x' - y'\|_1$ (see, e.g., [KN06, Lemma 3.1]).

Recall that N_ϵ is a vector of d random independent boolean values, each equal to one with probability $1/2-\epsilon/2$. Let η_0 be the uniform distribution over pairs $(x,y) \in \{0,1\}^d \times \{0,1\}^d$, and let η_1 be a distribution over pairs $(x,y) \in \{0,1\}^d \times \{0,1\}^d$ where x is random and $y = x + N_\epsilon$ for $\epsilon = 1 - R/\alpha d$. For $i \in \{0,1\}$, define μ_i as the distribution of (Gx, Gy) where (x,y) are picked from η_i. Since not all the pairs (X,Y) in the support of μ_i are legal (i.e., some pairs are not far or close, respectively), we define $\tilde{\mu}_i$ as the restriction of μ_i to legal pairs. Namely, $\tilde{\mu}_0$ is the distribution μ_0 conditioned on the fact that $\mathrm{EMD}(X,Y) > R$; similarly we define μ_1.

We need show that, for each $i \in \{0,1\}$, the statistical distance between μ_i and $\tilde{\mu}_i$ is $2^{-\Omega(d/\alpha)}$, and, hence, by switching between the two we do not loose anything. For (x,y) drawn from η_1, by Chernoff bound (Fact 2.4.4), with probability at least $1-2^{-\Omega(d/\alpha)}$, we have $\|x-y\|_1 \leq 2d(1/2-\epsilon/2) = R/\alpha$, implying that $\mathrm{EMD}(X,Y) \leq R/\alpha$, i.e. $(X,Y) = (Gx, Gy)$ is a close pair. Similarly, for (x,y) drawn from η_0, we can prove that for every $x' \in Gx$, the probability that $\|y - x'\|_1 > d/100$ is at least $1 - 2^{-\Omega(d)}$. Indeed, for every $x' \in Gx$, the number of points at distance $\leq d/100$ from x' is at most by $\binom{d}{d/100} \leq 2^{d/10}$. Thus, with probability at least $1 - s \cdot 2^{d/10}/2^d \geq 1 - 2^{-\Omega(d)}$, for all $x' \in Gx$ we have $\|y - x'\|_1 > d/100$, implying that $\mathrm{EMD}(X,Y) > R$, i.e., $(X,Y) = (Gx, Gy)$ is a far pair.

Now we proceed to proving that the constructed distribution satisfies inequality (7.4) for all boolean functions $\mathcal{H}^A, \mathcal{H}^B$ on subsets of $\{0,1\}^d$ of size s. For this, we fix some functions $\mathcal{H}^A, \mathcal{H}^B$ and define $f, g : \{0,1\}^d \to \{0,1\}$ as the extensions of $\mathcal{H}^A$ and $\mathcal{H}^B$ to the $\{0,1\}^d$ cube in the natural way: $f(x) \triangleq \mathcal{H}^A(Gx)$ and $g(x) \triangleq \mathcal{H}^B(Gx)$. By the definition of G, for all $a \in C^\perp$ we have that $f(x) = f(x+a)$ and $g(x) = g(x+a)$. Furthermore, for all $i \in \{0,1\}$, we have that $\Pr_{\eta_i}[f(x) = g(y)] = \Pr_{\mu_i}\left[\mathcal{H}^A(Gx) = \mathcal{H}^B(Gy)\right]$. At this moment, we can bound the distinguishing power of $\mathcal{H}^A$ and $\mathcal{H}^B$ as follows:

$$\begin{aligned}
&\Pr_{(X,Y)\leftarrow\tilde{\mu}_0}[\mathcal{H}^A(X)\neq\mathcal{H}^B(Y)] - \Pr_{(X,Y)\leftarrow\tilde{\mu}_1}[\mathcal{H}^A(X)\neq\mathcal{H}^B(Y)] \\
&\leq \Pr_{(X,Y)\leftarrow\mu_0}[\mathcal{H}^A(X)\neq\mathcal{H}^B(Y)] - \Pr_{(X,Y)\leftarrow\mu_1}[\mathcal{H}^A(X)\neq\mathcal{H}^B(Y)] + \mathrm{TV}(\mu_0,\tilde{\mu}_0) + \mathrm{TV}(\mu_1,\tilde{\mu}_1) \\
&\leq \Pr_{(x,y)\leftarrow\eta_0}[f(x)\neq g(y)] - \Pr_{(x,y)\leftarrow\eta_1}[f(x)\neq g(y)] + 2^{-\Omega(d/\alpha)}. \\
&= \mathbb{E}_{(x,y)\leftarrow\eta_0}\left[(f(x)-g(y))^2\right] - \mathbb{E}_{(x,y)\leftarrow\eta_1}\left[(f(x)-g(y))^2\right] + 2^{-\Omega(d/\alpha)}, \\
&= \mathbb{E}_{(x,y)\leftarrow\eta_0}\left[f(x)^2-2f(x)g(y)+g(y)^2\right] - \mathbb{E}_{(x,y)\leftarrow\eta_1}\left[f(x)^2-2f(x)g(y)+g(y)^2\right] + 2^{-\Omega(d/\alpha)}, \\
&= 2\mathbb{E}_{(x,y)\leftarrow\eta_1}\left[f(x)g(y)\right] - 2\mathbb{E}_{(x,y)\leftarrow\eta_0}\left[f(x)g(y)\right] + 2^{-\Omega(d/\alpha)}, \qquad (7.5)
\end{aligned}$$

where we used the bound on statistical distance between μ_i and $\tilde{\mu}_i$ for $i\in\{0,1\}$, the definition of f,g, and the fact that η_0 and η_1 have uniform marginals.

We now use the Fourier properties of the functions f and g to bound the expectation of $f(x)g(y)$ under the distributions η_0 and η_1. Suppose $\hat{f}_S$ are the Fourier coefficients of f, for $S\subseteq[d]$; define $\hat{g}_S$ similarly. Note that $\hat{f}_\emptyset = \mathbb{E}_x\left[f(x)\right]$ and $\hat{g}_\emptyset = \mathbb{E}_x\left[g(x)\right]$ for the uniform distribution on x. Thus, we have

$$\mathbb{E}_{(x,y)\leftarrow\eta_0}\left[f(x)g(y)\right] = \hat{f}_\emptyset\hat{g}_\emptyset. \qquad (7.6)$$

Now we compute the expectation for (x,y) drawn from η_1. Using Parseval's identity and Fact 7.1.2 for $p=2$, we have

$$\mathbb{E}_{\eta_1}\left[f(x)g(y)\right] = \mathbb{E}_x\left[f(x)\mathbb{E}_{N_\epsilon}\left[g(y+N_\epsilon)\right]\right] = \mathbb{E}_x\left[f(x)\mathbb{E}_{N_\epsilon}\left[(T_\epsilon g)(y)\right]\right] = \sum_{S\subseteq[d]}\hat{f}_S\widehat{(T_\epsilon g)}_S = \sum_{S\subseteq[d]}\hat{f}_S\hat{g}_S\epsilon^{|S|}. \qquad (7.7)$$

Now we use a crucial Fourier-analytic property of the functions f,g. Specifically, the condition that $f(x)=f(x+a)$ for all $a\in C^\perp$ implies that that $\hat{f}_S=0$ for all S with $0<|S|<w(C)$, where $w(C)\geq cr$ is the weight of the code C (see [KN06, Lemma 3.3]). Hence, in inequality (7.7), the sum is only over sets S such that $|S|\geq cd$, and thus we obtain that

$$\mathbb{E}_{\eta_1}\left[f(x)g(y)\right] \leq \hat{f}_\emptyset\hat{g}_\emptyset + \epsilon^{cd}\sum_{S\subseteq[d]}\hat{f}_S\hat{g}_S = \hat{f}_\emptyset\hat{g}_\emptyset + \epsilon^{cd}\mathbb{E}_{x,y\leftarrow\{0,1\}^d}\left[f(x)g(y)\right] \leq \hat{f}_\emptyset\hat{g}_\emptyset + \epsilon^{cd}, \qquad (7.8)$$

where the equality uses the Parseval's identity.

Finally, plugging-in Eqn. (7.6) and Eqn. (7.8) into Eqn. (7.5), as well as $\epsilon = 1-R/\alpha d$, we obtain that, for any boolean functions $\mathcal{H}^A,\mathcal{H}^B$, we have

$$\begin{aligned}
\Pr_{(X,Y)\leftarrow\tilde{\mu}_0}[\mathcal{H}^A(X)\neq\mathcal{H}^B(Y)] - &\Pr_{(X,Y)\leftarrow\tilde{\mu}_1}[\mathcal{H}^A(X)\neq\mathcal{H}^B(Y)] \\
&\leq 2(\hat{f}_\emptyset\hat{g}_\emptyset + \epsilon^{cd}) - 2\hat{f}_\emptyset\hat{g}_\emptyset + 2^{-\Omega(d/\alpha)} \\
&= 2(1-R/\alpha d)^{cd} + 2^{-\Omega(d/\alpha)} \\
&\leq 2^{-\Omega(d/\alpha)}.
\end{aligned}$$

We have completed proving inequality (7.4) and thus the proof of Theorem 7.0.9. □

7.4 Ulam and Edit Distances Lower Bound

We now proceed to proving Theorem 7.0.6, whose proof will take the rest of the chapter. We focus on the lower bound for the Ulam distance, i.e., on lower bounding $\mathrm{CC}^{\mathrm{Ulam}_d}$. Once we establish this, the lower bound for edit distance on binary strings follows immediately by the reduction from Ulam distance to edit distance, namely Lemma 2.1.1 from Chapter 2.

Fix the values of d and R, and let us use the alphabet $\Sigma = \mathbb{Z}_p$ for p sufficiently large so that a random string from Σ^d is non-repetitive with high probability (e.g., it suffices to set $p = d^3$). As before, we denote our hard distribution by $\tilde{\mu} = \frac{\tilde{\mu}_0 + \tilde{\mu}_1}{2}$, where $\tilde{\mu}_0$ will be a distribution over far pairs of strings (x, y) and $\tilde{\mu}_1$ will be a distribution over close pairs (x, y), i.e., $\mathrm{ed}(x, y) > R$ and $\mathrm{ed}(x, y) \leq R/\alpha$, respectively.

We will follow the steps outlined in Section 7.1 and eventually put all the pieces together in Section 7.4.2. Our general approach to proving the theorem uses just a few simple properties of the hard distribution, which we will specify along the way. To differentiate the underlying technique from the specifics of our hard distribution, we describe the hard distribution and prove its required properties separately, in Section 7.5.

Applying Lemma 7.2.1, we need to just disprove the inequality (7.2) for all boolean functions $\mathcal{H}^A, \mathcal{H}^B$. Thus, for the rest of the section, we assume the existence of boolean functions $\mathcal{H}^A, \mathcal{H}^B$ for the sake of contradiction.

7.4.1 λ-Tests

In this section, we provide a method to lower bound the advantage achieved by the boolean functions $\mathcal{H}^A, \mathcal{H}^B$, by relating it to a certain statistical property of the hard distribution $\tilde{\mu}$. Our hard distribution $\tilde{\mu} = \frac{\tilde{\mu}_0 + \tilde{\mu}_1}{2}$ will have a specific generic construction that we describe next. For each $t \in \{0, 1\}$, the distribution $\tilde{\mu}_t$ is formed via a small modification of another distribution μ_t, which is easier to analyze (due to certain independencies), but might (rarely) produce invalid inputs. Specifically, each $\tilde{\mu}_t$ is the distribution μ_t conditioned on the fact that the pair $(x, y) \in \mu_t$ is valid in the sense that x and y are both permutations and the pair (x, y) is respectively a far (when $t = 0$) or a close (when $t = 1$) pair. We analyze below the distributions μ_0 and μ_1 (specifically, in Lemma 7.4.4). For completeness, we mention that, in the next section, we show that this analysis extends to distributions $\tilde{\mu}_0$ and $\tilde{\mu}_1$ using the fact that $\tilde{\mu}_0$ and $\tilde{\mu}_1$ are statistically very close to distributions μ_0 and μ_1 respectively.

The distribution μ_t consists of pairs (x, y) chosen as follows: $x \in \mathbb{Z}_p^d$ is chosen uniformly at random, and y is constructed from x in two steps. In the first step, let $z \triangleq x + N_\rho$, where N_ρ, defined in the preliminaries, is noise of rate $\rho \in (0, 1)$, independent of t. In the second step, y is obtained from z by permuting the coordinates of z according to a distribution $\mathcal{D}_t$. Formally, $\mathcal{D}_t$ is a distribution over permutation operations, where a *permutation operation* is a function $\pi : \mathbb{Z}_p^d \to \mathbb{Z}_p^d$ for which there exists a permutation $\hat{\pi} : [d] \to [d]$ such that $\pi(x) \equiv (x_{\hat{\pi}(1)}, \ldots x_{\hat{\pi}(d)})$. We will require that $\mathcal{D}_t$ is *symmetric* in the sense that, for every π, the permutation operations π and π^{-1} are equi-probable (in it). Notice that y has the same marginal distribution as x, i.e. uniform over $\mathbb{Z}_p^d$.

We now quantify the "difference" between the distributions $\mathcal{D}_0, \mathcal{D}_1$ from the perspective of what we call *λ-tests*. For $\lambda \in [d]$, we define a *λ-test* to be a vector $u \in \mathbb{Z}_p^d$ with precisely λ non-zero entries, i.e., $\mathrm{wt}(u) = \lambda$. For a distribution $\mathcal{D}_t$ and $\lambda \in [d]$, let the matrix $A^{(t,\lambda)}$ be the transition matrix of a Markov chain whose states are all the λ-tests, and whose transitions are according to $\mathcal{D}_t$, i.e., at a λ-test u, the process picks $\pi \in \mathcal{D}_t$ and moves to state $\pi(u)$ (which is also a λ-test). In other words, a row corresponding to u in $A^{(t,\lambda)}$ is a vector, that has, for every λ-test w, a coordinate of value $\Pr_{\pi \in \mathcal{D}_t}[\pi(u) = w]$. We denote this row by $A_u^{(t,\lambda)}$. Note that the matrix $A^{(t,\lambda)}$ is symmetric (since $\mathcal{D}_t$ is symmetric) and thus it is doubly-stochastic.

Definition 7.4.1. *The* λ-test distinguishability *of $\mathcal{D}_0, \mathcal{D}_1$, denoted Δ_λ, is the maximum, over all λ-tests u, of the total variation distance between the distributions $A_u^{(0,\lambda)}$ and $A_u^{(1,\lambda)}$.*

We can also write Δ_λ using matrix norms.

Definition 7.4.2. *For matrix $B \in M_{n,n}(\mathbb{R})$ and $p \in [1, \infty]$, the* p-norm *of B is defined by* $\|B\|_p = \max\{\|Bv\|_p : \ v \in \mathbb{C}^n, \|v\|_p = 1\}$.

In particular, $\|B\|_\infty = \max_{i \in [n]} \sum_{j \in [n]} |B_{ij}|$ *for all* $B \in M_{n,n}(\mathbb{R})$.

Fact 7.4.3. $\Delta_\lambda = \|A^{(0,\lambda)} - A^{(1,\lambda)}\|_\infty / 2$.

Later (in Fact 7.4.8) we shall use known inequalities between different matrix norms (in particular ℓ_∞ and ℓ_2).

The following lemma bounds the advantage achieved by $\mathcal{H}^A, \mathcal{H}^B$ in terms of the λ-test distinguishability Δ_λ of distributions $\mathcal{D}_0$ and $\mathcal{D}_1$ for any pair of distributions $\mathcal{D}_0, \mathcal{D}_1$. Note that we have not yet specified the distributions $\mathcal{D}_0$ and $\mathcal{D}_1$ themselves. We will specify the distributions $\mathcal{D}_0$ and $\mathcal{D}_1$ in Section 7.5, thus completing the definition of the hard distribution $\tilde{\mu}$.

Lemma 7.4.4. *Consider $\mathcal{H}^A, \mathcal{H}^B : \mathbb{Z}_p^d \to \{-1, +1\}$ and $\rho \in (0, 1)$. If each μ_t, for $t \in \{0, 1\}$, is defined as above from a symmetric distributions $\mathcal{D}_t$ over permutation operations, then*

$$\Pr_{\mu_0}[\mathcal{H}^A(x) \neq \mathcal{H}^B(y)] - \Pr_{\mu_1}[\mathcal{H}^A(x) \neq \mathcal{H}^B(y)] \leq \max_{\lambda \in [d]} \Delta_\lambda \rho^\lambda.$$

Proof. For $t \in \{0, 1\}$, define $C^{(t)} \triangleq \mathbb{E}_{\mu_t}\left[\mathcal{H}^A(x)\mathcal{H}^B(y)\right]$ to be the *correlation* between the two boolean functions. Note that, $\Pr_{\mu_t}[\mathcal{H}^A(x) \neq \mathcal{H}^B(y)] = \frac{1}{4}\mathbb{E}_{\mu_t}\left[\mathcal{H}^A(x) - \mathcal{H}^B(y)\right]^2 = 1/2 - C^{(t)}/2$. Thus,

$$\Pr_{\mu_0}[\mathcal{H}^A(x) \neq \mathcal{H}^B(y)] - \Pr_{\mu_1}[\mathcal{H}^A(x) \neq \mathcal{H}^B(y)] = \frac{C^{(1)} - C^{(0)}}{2}.$$

We will show that $C^{(1)} - C^{(0)} \leq 2 \max_{\lambda \in [d]} \Delta_\lambda \rho^\lambda$. For this purpose, it is more convenient to express each $C^{(t)}$ in terms of the Fourier coefficients of $\mathcal{H}^A$ and $\mathcal{H}^B$. Recall that μ_t is generated by picking a random x, and constructing y from x by adding to it the noise N_ρ and then applying a random permutation drawn from $\mathcal{D}_t$, namely, $y = \pi(x + N_\rho)$, where $\pi \in \mathcal{D}_t$. Let $\mu_t|x$ denote the distribution μ_t conditioned on the value of x. Thus,

$$\mathbb{E}_{\mu_t}\left[\mathcal{H}^A(x)\mathcal{H}^B(y)\right] = \mathbb{E}_{x \in \mathbb{Z}_p^d}\left[\mathcal{H}^A(x) \cdot \mathbb{E}_{\mu_t|x}\left[\mathcal{H}^B(y)\right]\right]$$

Define $f^{(t)}(x) \triangleq \mathbb{E}_{\mu_t|x}\left[\mathcal{H}^B(y)\right]$. Then

$$f^{(t)}(x) = \mathbb{E}_{N_\rho}\left[\mathbb{E}_{\pi \leftarrow \mathcal{D}_t}\left[\mathcal{H}^B(\pi(x + N_\rho))\right]\right].$$

Since $C^{(t)} = \mathbb{E}_x\left[\mathcal{H}^A(x) f^{(t)}(x)\right]$, we can switch to the Fourier basis by applying Parseval's identity, and get

$$C^{(t)} = \sum_{u \in \mathbb{Z}_p^d} \widehat{(\mathcal{H}^A)}_u \overline{\widehat{(f^{(t)})}}_u, \tag{7.9}$$

where $\widehat{(\mathcal{H}^A)}_u$ and $\widehat{(f^{(t)})}_u$ are the Fourier coefficients of $\mathcal{H}^A$ and $f^{(t)}$ respectively.

The next proposition, which we shall prove shortly, expresses the level λ Fourier coefficients of $f^{(t)}$ in terms of those of $\mathcal{H}^B$. Let $\left(\widehat{(f^{(t)})}_u\right)_{u:\mathrm{wt}(u)=\lambda}$ be the vector of the Fourier coefficients of $f^{(t)}$ indexed by u's of weight $\mathrm{wt}(u) = \lambda$. Define $\left(\widehat{(\mathcal{H}^B)}_u\right)_{u:\mathrm{wt}(u)=\lambda}$ similarly.

Proposition 7.4.5. *For all $\lambda \in [d]$ and $\mathcal{H}^B : \mathbb{Z}_p^d \to \mathbb{C}$,*

$$\left(\widehat{(f^{(t)})}_u\right)_{u:\mathrm{wt}(u)=\lambda} = \rho^\lambda A^{(t,\lambda)} \cdot \left(\widehat{(\mathcal{H}^B)}_u\right)_{u:\mathrm{wt}(u)=\lambda} \tag{7.10}$$

This proposition naturally leads us to break each $C^{(t)}$ into the terms corresponding to each Fourier level λ. Define the λ^{th}*-correlation* to be

$$C_\lambda^{(t)} \triangleq \sum_{u\in\mathbb{Z}_p^d:\mathrm{wt}(u)=\lambda} \widehat{(\mathcal{H}^A)}_u \overline{\widehat{(f^{(t)})}_u}. \tag{7.11}$$

Then, $C^{(1)} - C^{(0)} = \sum_{\lambda=0}^d \left(C_\lambda^{(1)} - C_\lambda^{(0)}\right)$. We can now bound each $C_\lambda^{(1)} - C_\lambda^{(0)}$ in terms of Δ_λ and ρ.

Let $\omega_\lambda^A = \left\|\left(\widehat{(\mathcal{H}^A)}_u\right)_{u:\mathrm{wt}(u)=\lambda}\right\|_2$ be the ℓ_2-weight of the level λ Fourier coefficients of $\mathcal{H}^A$, and define similarly ω_λ^B. By Parseval's identity, $\sum_{\lambda=0}^d \left(\omega_\lambda^A\right)^2 = \mathbb{E}_x\left[\mathcal{H}^A(x)\cdot\overline{\mathcal{H}^A(x)}\right] = 1$, and similarly $\sum_{\lambda=0}^d \left(\omega_\lambda^B\right)^2 = 1$.

Proposition 7.4.6. *For all $\lambda \in [d]$,*

$$C_\lambda^{(1)} - C_\lambda^{(0)} \leq 2\Delta_\lambda\rho^\lambda \cdot \omega_\lambda^A\omega_\lambda^B.$$

We will prove the proposition shortly by a straightforward calculation. In addition, $C_0^{(1)} = C_0^{(0)}$ because the 0-th level Fourier coefficient of $f^{(t)}$ equals $\mathbb{E}_{x\in\mathbb{Z}_p^d}\left[f^{(t)}(x)\right] = \mathbb{E}_{y\in\mathbb{Z}_p^d}\left[\mathcal{H}^B(y)\right]$, which does not depend on $t \in \{0,1\}$. Given the above proposition, we thus have

$$\begin{aligned} C^{(1)} - C^{(0)} &= \sum_{\lambda=0}^d \left(C_\lambda^{(1)} - C_\lambda^{(0)}\right) \leq \sum_{\lambda=1}^d 2\Delta_\lambda\rho^\lambda \cdot \omega_\lambda^A\omega_\lambda^B \\ &\leq \sum_{\lambda=1}^d 2\Delta_\lambda\rho^\lambda \cdot \frac{\left(\omega_\lambda^A\right)^2 + \left(\omega_\lambda^B\right)^2}{2} \leq 2\max_{\lambda\in[d]} \Delta_\lambda\rho^\lambda, \end{aligned}$$

where we used the geometric–arithmetic mean inequality. This finishes the proof of Lemma 7.4.4. $\square$

It remains to prove Propositions 7.4.5 and 7.4.6.

Proof of Proposition 7.4.5. Define a new function $g^{(t)} : \mathbb{Z}_p^d \to \mathbb{R}$ as

$$g^{(t)}(z) \triangleq \mathbb{E}_{\pi\leftarrow\mathcal{D}_t}\left[\mathcal{H}^B(\pi(z))\right].$$

Then $f^{(t)} = T_\rho g^{(t)}$, and thus $\widehat{(f^{(t)})}_u = \widehat{(g^{(t)})}_u \cdot \rho^{\mathrm{wt}(u)}$ for all $u \in \mathbb{Z}_p^d$ (by Fact 7.1.2). It remains to prove that

$$\left(\widehat{(g^{(t)})}_u\right)_{u:\mathrm{wt}(u)=\lambda} = A^{(t,\lambda)} \cdot \left(\widehat{(\mathcal{H}^B)}_u\right)_{u:\mathrm{wt}(u)=\lambda} \tag{7.12}$$

Similarly to the operator T_ρ, we define the operator $\mathcal{O}_t$ as $(\mathcal{O}_t \mathcal{H}^B)(x) \triangleq \mathbb{E}_{\pi \leftarrow \mathcal{D}_t}\left[\mathcal{H}^B(\pi(x))\right]$. Since $g^{(t)} = \mathcal{O}_t \mathcal{H}^B$, we proceed to analyze how the operator $\mathcal{O}_t$ works on the Fourier coefficients of a function $\mathcal{H}^B$.

Fact 7.4.7. *For a permutation operation π, define $\mathcal{P}_\pi$ to be an operator on functions $\psi : \mathbb{Z}_p^d \to \mathbb{R}$, given by $(\mathcal{P}_\pi \psi)(x) \triangleq \psi(\pi(x))$. Then, $\widehat{(\mathcal{P}_\pi \psi)}_u = \hat{\psi}_{\pi(u)}$.*

Now, the operator $\mathcal{O}_t$ defined earlier is simply a convex combination of several $\mathcal{P}_\pi$, where π is drawn from $\mathcal{D}_t$. Thus, with the above fact, for every $u \in \mathbb{Z}_p^d$,

$$\widehat{(g^{(t)})}_u = \widehat{(\mathcal{O}_t \mathcal{H}^B)}_u = \mathbb{E}_{\pi \leftarrow \mathcal{D}_t}\left[\widehat{(\mathcal{H}^B)}_{\pi(u)}\right]. \tag{7.13}$$

Consequently, the vector of level λ Fourier coefficients of $g^{(t)}$ can be written as a product of the matrix $A^{(t,\lambda)}$ and the vector of the (same) level λ Fourier coefficients of $\mathcal{H}^B$, which proves Proposition 7.4.5. □

We will need the following fact for the proof of Proposition 7.4.6. Recall that $\|A\|_p$ denotes the p-norm of such a matrix A, as per Definition 7.4.2.

Fact 7.4.8. *Let $B \in M_{n,n}(\mathbb{R})$ be a symmetric matrix. Then, $\|B\|_2 \le \|B\|_\infty$.*

Proof. It is known that $\|B\|_1 = \max_{j \in [n]} \sum_{i \in n} |B_{ij}|$ and $\|B\|_\infty = \max_{i \in [n]} \sum_{j \in [n]} |B_{ij}|$, and since B is symmetric, these two norms are equal. By Riesz-Thorin interpolation theorem, $\|B\|_2 \le \max\{\|B\|_1, \|B\|_\infty\} = \|B\|_\infty$. (The Riesz-Thorin interpolation theorem states that for every $1 \le p < q < r \le \infty$ and a real matrix A, we have $\|A\|_q \le \max\{\|A\|_p, \|A\|_r\}$.) □

Proof of Proposition 7.4.6. For every λ, the matrix $A^{(t,\lambda)}$ is symmetric, and so is $A^{(1,\lambda)} - A^{(0,\lambda)}$. Thus,

$$\begin{aligned}
C_\lambda^{(1)} - C_\lambda^{(0)} &= \sum_{u \in \mathbb{Z}_p^d : \mathrm{wt}(u)=\lambda} \widehat{(\mathcal{H}^A)}_u \cdot \left(\overline{\widehat{(f^{(1)})}_u} - \overline{\widehat{(f^{(0)})}_u}\right) \\
&\le \left\|\left(\widehat{(\mathcal{H}^A)}_u\right)_{u:\mathrm{wt}(u)=\lambda}\right\|_2 \cdot \left\|\left(\overline{\widehat{(f^{(1)})}_u} - \overline{\widehat{(f^{(0)})}_u}\right)_{u:\mathrm{wt}(u)=\lambda}\right\|_2 \\
&= \omega_\lambda^A \cdot \left\|\rho^\lambda \left(A^{(1,\lambda)} - A^{(0,\lambda)}\right)\left(\overline{\widehat{(\mathcal{H}^B)}_u}\right)_{u:\mathrm{wt}(u)=\lambda}\right\|_2 \\
&\le \rho^\lambda \cdot \omega_\lambda^A \cdot \left\|A^{(1,\lambda)} - A^{(0,\lambda)}\right\|_2 \left\|\left(\widehat{(\mathcal{H}^B)}_u\right)_{u:\mathrm{wt}(u)=\lambda}\right\|_2 \\
&\le \rho^\lambda \cdot \omega_\lambda^A \omega_\lambda^B \cdot \left\|A^{(1,\lambda)} - A^{(0,\lambda)}\right\|_\infty \\
&= 2\Delta_\lambda \cdot \rho^\lambda \cdot \omega_\lambda^A \omega_\lambda^B;
\end{aligned}$$

where we used Eqn. (7.11), Cauchy-Schwarz, Proposition 7.4.5, Definition 7.4.2, Fact 7.4.8, and Definition 7.4.3, respectively. □

7.4.2 Proof of Theorem 7.0.6

We proceed to proving Theorem 7.0.6, using the machinery developed in Sections 7.2 and 7.4.1. Recall that we still need to exhibit a suitable hard distribution. We outlined the construction of our hard distribution in Section 7.4.1 ; the construction relies on two distributions $\mathcal{D}_0$ and $\mathcal{D}_1$ which were not specified. The next lemma asserts that the desired hard distribution exists. More precisely, it asserts that it can be constructed to satisfy the required properties, such as a small λ-test distinguishability.

Lemma 7.4.9 (Hard Distribution)**.** *There exist constants $\theta, c_1, c_2, d_0 > 0$, such that for all $d > d_0$, $p > d^3$, $1 < \alpha \le O(\frac{\log d}{\log\log d})$, $d^{0.1} \le R \le d^{0.49}$, there exist symmetric distributions $\mathcal{D}_0^*$ and $\mathcal{D}_1^*$ over permutation operations on $\mathbb{Z}_p^d$ (as defined in Section 7.4.1) with the following guarantees.*

(a). For all $\lambda \ge 1$, the λ-test distinguishability of $\mathcal{D}_0^*$ and $\mathcal{D}_1^*$ is $\Delta_\lambda \le c_1 \cdot \lambda \frac{\log\alpha}{\log d} \cdot \frac{R}{d}$.

(b). Define each distribution μ_t from $\mathcal{D}_t^*$ as described in Section 7.4.1, setting $\rho = 1 - \theta\frac{R/\alpha}{d}$. Define the distribution $\tilde{\mu}_t$ to be the restriction (i.e. conditioning) of μ_t to the event that the sampled pair $(x, y) \in \mu_t$ is legal, in the sense that $x, y \in \mathbb{Z}_p^d$ are permutations and are respectively a far pair (for $t = 0$) or a close pair (for $t = 1$). Then for each $t \in \{0, 1\}$, the total variation distance between $\tilde{\mu}_t$ and μ_t is at most d^{-c_2}.

We prove this lemma separately in Section 7.5, where we include a full description of $\mathcal{D}_0^*$ and $\mathcal{D}_1^*$. Here, we use the lemma to complete the proof of the main theorem.

Proof of Theorem 7.0.6. First, consider the hard distribution given by Lemma 7.4.9. Next, by Lemma 7.2.1, there must exist functions $\mathcal{H}^A, \mathcal{H}^B$ such that

$$\Pr_{\tilde{\mu}_0}[\mathcal{H}^A(x) \ne \mathcal{H}^B(y)] - \Pr_{\tilde{\mu}_1}[\mathcal{H}^A(x) \ne \mathcal{H}^B(y)] \ge \tfrac{1}{3} \cdot 2^{-\mathrm{CC}_{\alpha,R}^{\mathrm{Ulam}_d}}.$$

Applying Lemma 7.4.4 to the distributions μ_0, μ_1, and using the fact that $\tilde{\mu}_0$ and $\tilde{\mu}_1$, respectively are statistically close to μ_0 and μ_1 (Lemma 7.4.9(b)), we deduce that

$$\Pr_{\tilde{\mu}_0}[\mathcal{H}^A(x) \ne \mathcal{H}^B(y)] - \Pr_{\tilde{\mu}_1}[\mathcal{H}^A(x) \ne \mathcal{H}^B(y)] \le \max_{\lambda\in[d]} \Delta_\lambda \rho^\lambda + d^{-c_2}.$$

Combining the two inequalities above and plugging in the upper bound on Δ_λ and the value of ρ from Lemma 7.4.9(a), we have

$$\begin{aligned}
\tfrac{1}{3} \cdot 2^{-\mathrm{CC}_{\alpha,R}^{\mathrm{Ulam}_d}} &\le \max_{\lambda\in[d]} \left[c_1 \cdot \lambda \tfrac{\log\alpha}{\log d} \cdot \tfrac{R}{d} \cdot \left(1 - \theta\tfrac{R/\alpha}{d}\right)^\lambda \right] + d^{-c_2} \\
&\le \tfrac{c_1}{\theta} \cdot \alpha \cdot \frac{\log\alpha}{\log d} \cdot \max_{x\ge 0} x \cdot e^{-x} + d^{-c_2} \\
&= O\left(\tfrac{\alpha\log\alpha}{\log d}\right),
\end{aligned}$$

which concludes the proof of Theorem 7.0.6. ∎

7.5 The Hard Distribution for the Ulam Distance

In this section we prove Lemma 7.4.9. Namely, we construct our hard distribution for the Ulam distance, and show the required properties for Theorem 7.0.6 to hold.

We start by giving the detailed construction of our hard distribution $\tilde{\mu} = \frac{\tilde{\mu}_0+\tilde{\mu}_1}{2}$. Then, in Sections 7.5.1 and 7.5.2 we prove respectively λ-test indistinguishability (part (a)) and statistical closeness (part (b)) properties of the hard distribution.

The hard distribution construction follows the outline given in Section 7.4.1. We first specify the distributions $\mathcal{D}_0^*, \mathcal{D}_1^*$ over permutation operators, which forms the bulk of the construction. Once these distributions are specified, we obtain the intermediary distributions μ_0 and μ_1 as already described in Section 7.4.1. We finalize the description by constructing $\tilde{\mu}_t$ from μ_t, for each $t \in \{0,1\}$, by conditioning on the pair $(x,y) \in \mu_t$ being a legal pair, namely that $x, y \in \mathbb{Z}_p^d$ are permutations and are respectively a far pair (for $t=0$) or a close pair (for $t=1$).

Fix $\epsilon_0 \triangleq 1/2$ and select $\epsilon_1 = \Theta(\frac{1}{\alpha})$ as follows. Let $\beta \triangleq \frac{1-\epsilon_1}{1-\epsilon_0} = 2(1-\epsilon_1)$, and $\xi_1 \triangleq \lceil \log_2(C_1\alpha) \rceil$, for a sufficiently large constant $C_1 > 0$ (in particular $C_1 = 805$ will suffice). Let ϵ_1 be the solution to the equation $(1-\epsilon_1) = \epsilon_1 \beta^{\xi_1}$ satisfying $\epsilon_1 \leq \frac{2}{C_1\alpha}$. The existence of ϵ_1 follows from the following claim, whose proof is deferred to the end of the construction.

Claim 7.5.1. *Let $\alpha > 1$ and $C_1 > 1$ be sufficiently large. Then there exists ϵ_1 with $\frac{1}{3C_1\alpha} < \epsilon_1 \leq \frac{2}{C_1\alpha}$ such that $(1-\epsilon_1) = \epsilon_1(2(1-\epsilon_1))^{\xi_1}$, where $\xi_1 = \lceil \log_2(C_1\alpha) \rceil$.*

We thus have, by construction,

$$\epsilon_0 = (1-\epsilon_0) = (1-\epsilon_1)\beta^{-1} = \epsilon_1 \beta^{\xi_1 - 1}. \tag{7.14}$$

For each $t \in \{0,1\}$, we define the distribution μ_t over (x,y) such that $\mathrm{ed}(x,y)$ is almost surely $\Theta(\epsilon_t R)$. Choose $x \in \Sigma^d = \mathbb{Z}_p^d$ uniformly at random. Then set $z \triangleq x + N_\rho$ where $N_\rho \in \mathbb{Z}_p^d$ is a noise of rate $\rho \triangleq 1 - \epsilon_1 R/d$ (i.e., each position is randomized with probability $1-\rho = \epsilon_1 R/d$). We shall obtain y from z, by applying a number of random rotation operations, each picked independently from a specific distribution. We use the following notation:

- $m \triangleq 0.01 \cdot \log_\beta d = \Theta(\log d)$ is the number of possible lengths of a rotation operation;
- $L_{\min} \triangleq d^{0.01}$ determines the minimum length of a rotation operation (modulo a factor of β);
- $w \triangleq C_2 \cdot \frac{R}{m \cdot L_{\min}}$ is the number of rotation operations that we apply, for a sufficiently large constant $C_2 > 0$ to be determined later (in Section 7.5.2).

Generate a sequence $(r_1, r_2, \ldots, r_w)$ of w rotations by picking each r_i i.i.d. according to the following distribution $\mathcal{D}_t^{\mathrm{rot}}$:

1. Pick $l_i \in [m]$ randomly so that $\Pr[l_i = l] = \frac{\beta^{-l}}{\zeta}$ for each $l \in [m]$, where $\zeta = \sum_{l=1}^m \beta^{-l}$ is the normalization constant.
2. Pick a starting position $s_i \in [d]$ uniformly at random, and rotate the block that starts at position s_i and has length (with wrap-around) $L_i = \beta^{l_i} L_{\min}$ by $\epsilon_t L_i$ positions, either to the right or to the left, at random. We choose r_i at random from the set $\left\{ (\tilde{R}_{s,L_i})^{\epsilon_t L_i} \mid s \in [d], \tilde{R} \in \{\overrightarrow{R}, \overleftarrow{R}\} \right\}$.

   We note that $(\tilde{R}_{s,L_i})^{\epsilon_t L_i}$ is not well defined when $\epsilon_t L_i$ or L_i are not integers. Overloading the notation, we define $(\overrightarrow{R}_{s,L_i})^{\epsilon_t L_i}$ for non-integer $\epsilon_t L_i, L_i$ as follows. Let B_1

be the block that starts at position s_i and has length $\lfloor (1-\epsilon_t)L_i \rfloor$, and let B_2 be the block immediately following B_1 of length $\lfloor \epsilon_t L_i \rfloor$, i.e.

$$B_1 = [s : s + \lfloor (1-\epsilon_t)L_i \rfloor - 1], \quad B_2 = [s + \lfloor (1-\epsilon_t)L_i \rfloor : s + \lfloor (1-\epsilon_t)L_i \rfloor + \lfloor \epsilon_t L_i \rfloor - 1].$$

Then, $(\overrightarrow{R}_{s,L_i})^{\epsilon_t L_i}$ swaps blocks B_1 and B_2. We define $(\overleftarrow{R}_{s,L_i})^{\epsilon_t L_i}$ similarly.

To obtain y, we apply to $z = x + N_\rho$ the sequence of rotations $r_1, \ldots, r_w$, i.e.,

$$y \triangleq r_w(r_{w-1}(\ldots r_1(z)\ldots)) = (r_w \circ \ldots \circ r_2 \circ r_1)(x + N_\rho).$$

In the language of Section 7.4.1, the distribution $\mathcal{D}_t^*$ of permutation operations is simply the distribution of $\pi = r_w \circ r_{w-1} \circ \ldots \circ r_1$, where $r_1, \ldots r_w$ are drawn independently from $\mathcal{D}_t^{\text{rot}}$.

Intuitively, each rotation operation r_i, or more precisely its distribution $\mathcal{D}_t^{\text{rot}}$, is designed to achieve the following goal. Consider a position $j \in [d]$ and assume for simplicity $j \in [0.1d, 0.9d]$. Let the random variable $Z_{t,j} \in \mathbb{Z}$ be the displacement (change in position) of position j under a (random) rotation operation $r_i \in \mathcal{D}_t^{\text{rot}}$, i.e. $Z_{t,j} \in \mathbb{Z}$ is the unique value such that $r_i(e_j) = e_{j+Z_{t,j}}$, where e_k denotes the k-th standard basis vector. By construction, $Z_{t,j}$ is symmetric around 0, i.e. $\Pr[Z_{t,j} = k] = \Pr[Z_{t,j} = -k]$, and its distribution does not depend on j, i.e. $Z_{t,j}$ and $Z_{t,j'}$ have the same distribution (but they are correlated). Moreover, its support, i.e. values $k > 0$ with probability $\Pr[Z_{t,j} = k] > 0$, forms a geometric sequence (because the block length L has a geometric distribution). Let us now condition on the event that position j is included in the rotation block, i.e. $Z_{t,j} \neq 0$. Then the distribution of $Z_{t,j}$ is almost uniform over the support — this follows from the distribution of L and of s, and by Eqn. (7.14). Furthermore, the distributions of $Z_{0,j}$ and $Z_{1,j}$ (when we condition on them being nonzero) are almost identical, because their supports differ only at the boundaries, i.e. at the smallest and largest displacements, again due to Eqn. 7.14, and they are both almost uniform. We repeat the rotation operation many times in order to obtain a high concentration in the distance between y and z.

To finalize the construction, it remains to define $\tilde{\mu}_t$ for $t \in \{0,1\}$. We note that we cannot set $\tilde{\mu}_t$ to be exactly μ_t because the latter may sometimes generate pairs (x,y) that are not far or close respectively, or are not even permutations altogether. (x and y are not always permutations since each of the two strings is uniformly at random and may have a multiple occurrence of the same symbol.) We thus define $\tilde{\mu}_0$ to be the distribution μ_0 restricted to (i.e. conditioned on) pairs of permutations (x,y) with $\text{ed}(x,y) > R$, and similarly $\tilde{\mu}_1$ is the distribution μ_1 restricted to pairs of permutations with $\text{ed}(x,y) \leq R/\alpha$.

It only remains to prove Claim 7.5.1, namely that the desired ϵ_1 exists.

Proof of Claim 7.5.1. Define function $f(x) : [0,1] \to \mathbb{R}$ as $f(x) = x \cdot (1-x)^{\xi_1 - 1} 2^{\xi_1} - 1$. Note that ϵ_1 is the solution to the equation $f(x) = 0$. For $x = 1/(3C_1\alpha)$, $f(x) \leq \frac{1}{3C_1\alpha}(1 - \frac{1}{3C_1\alpha})^{\xi_1-1} \cdot 2^{\log_2(C_1\alpha)+1} - 1 < 0$. Similarly, for $x = \frac{2}{C_1\alpha}$, $f(x) \geq \frac{2}{C_1\alpha}(1 - \frac{2(\xi_1-1)}{C_1\alpha}) \cdot 2^{\log_2(C_1\alpha)} - 1 \geq 2(1 - \frac{2(\log_2(C_1\alpha)-1)}{C_1\alpha}) - 1 > 0$ provided C_1 is a sufficiently large constant. By the continuity of $f(x)$, there exists some $x \in [\frac{1}{3C_1\alpha}, \frac{2}{C_1\alpha}]$ satisfying $f(x) = 0$. $\square$

In the rest of this section we prove the two properties required from our hard distribution, stated in Lemma 7.4.9: that $\mathcal{D}_0^*$ and $\mathcal{D}_1^*$ have small λ-test distinguishability (Lemma 7.4.9 (a)), and that each $\tilde{\mu}_t$ is very close to μ_t, for both $t \in \{0,1\}$ (Lemma 7.4.9 (b)).

Here and throughout the big $O(\cdot)$ notation may hide dependence on constants used in the construction of the hard distribution, namely C_1 and C_2. Furthermore, although the parameters β and ζ are not constants (they depend on α), we can bound $1.5 < \beta < 2$, which guarantees that $\frac{1}{1-\beta^{-1}} \le O(1)$ and $\frac{1}{\zeta} \le \beta \le O(1)$.

7.5.1 λ-test indistinguishability

We prove Lemma 7.4.9 (a) via the following lemma.

Lemma 7.5.2. *Let Δ_λ be the λ-test distinguishability of $\mathcal{D}_0^*$ and $\mathcal{D}_1^*$. Then for all $\lambda \ge 1$, we have $\Delta_\lambda \le O\left(\lambda \frac{\log \alpha}{\log d} \cdot \frac{R}{d}\right)$.*

Proof. Fix a λ-test $u \in \mathbb{Z}_p^d$ and let $\delta_\lambda(u) = \max_{T \subseteq \mathbb{Z}_p^d} \left|\Pr[r^{(0)}(u) \in T] - \Pr[r^{(1)}(u) \in T]\right|$ be the total variation distance between the distributions $r^{(0)}(u)$ and $r^{(1)}(u)$, where $r^{(t)} \leftarrow \mathcal{D}_t^{\mathrm{rot}}$ for $t \in \{0, 1\}$. The heart of this lemma is the following bound, which we shall prove below:

$$\delta_\lambda(u) \le O\left(\lambda \log \alpha \cdot \frac{L_{\min}}{d}\right). \tag{7.15}$$

We shall also prove shortly the claim that $\Delta_\lambda \le w \cdot \max_u \delta_\lambda(u)$. The lemma then follows immediately from Eqn. (7.15) and this claim, by plugging the former into the latter and recalling $w = C_2 \cdot \frac{R}{m \cdot L_{\min}}$ is the number of rotation operations. Since $\lambda \frac{\log \alpha}{\log d} \cdot \frac{R}{d} > 1$ for $\lambda \ge d^{0.95}$, it actually suffices to prove (7.15) only for $\lambda < d^{0.95}$.

We now prove the above claim, that $\Delta_\lambda \le w \cdot \max_u \delta_\lambda(u)$, by induction. Let $v_i^t = r_i^{(t)}(r_{i-1}^{(t)}(\ldots r_1^{(t)}(u)\ldots))$ for $t \in \{0, 1\}$ and $i \in [w]$. We prove that, for any $T \subseteq \mathbb{Z}_p^d$, we have $|\Pr[v_i^0 \in T] - \Pr[v_i^1 \in T]| \le i \cdot \max_v \delta_\lambda(v)$. The base case $i = 1$ holds by the definition of δ_λ, and so we turn to the inductive step:

$$\begin{aligned}
\Pr[v_i^0 \in T] &= \sum_v \Pr[v_{i-1}^0 = v] \Pr[r_i^{(0)}(v) \in T] \\
&\le \sum_v \Pr[v_{i-1}^0 = v] \left(\Pr[r_i^{(1)}(v) \in T] + \delta_\lambda(v)\right) \\
&\le \max_v \delta_\lambda(v) + \sum_r \Pr[r_i^{(1)} = r] \Pr[r(v_{i-1}^0) \in T] \\
&\le \max_v \delta_\lambda(v) + \sum_r \Pr[r_i^{(1)} = r] \left(\Pr[r(v_{i-1}^1) \in T] + (i-1) \cdot \max_v \delta_\lambda(v)\right) \\
&= i \cdot \max_v \delta_\lambda(v) + \Pr[v_i^1 \in T].
\end{aligned}$$

Proving the same inequality with the roles of $t = 0$ and $t = 1$ reversed, we obtain that $\Delta_\lambda = \max_{T \subseteq \mathbb{Z}_p^d} |\Pr[v_w^0 \in T] - \Pr[v_w^1 \in T]| \le w \cdot \max_u \delta_\lambda(u)$.

In the rest of the proof of Lemma 7.5.2, we prove the bound (7.15). The proof consists of two parts. The first part proves the bound for $\lambda = 1$, and contains the main intuition why our distribution is hard. The second part builds on the first one to show the bound for general λ.

Part 1: $\lambda = 1$. We prove that $\delta_1(u) \le O(\log \alpha \cdot \frac{L_{\min}}{d})$ next. In this part, we shall assume that L and $\epsilon_t L$ are integers, deferring the full treatment of this technicality to the second part.

Since $\lambda = 1$, we have only one non-zero entry in u, say at position j. For $t \in \{0, 1\}$, let j_t be the random variable denoting the position of the symbol u_j in the vector $r^{(t)}(u)$ obtained by applying the random rotation $r^{(t)} \leftarrow \mathcal{D}_t^{\text{rot}}$ on u. Also, let Z_t be the displacement of j_t with respect to j on the cycle $\mathbb{Z}_p$, and namely $Z_t = (j_t - j + d/2)(\text{mod } d) - d/2$ (where the addition/subtraction of $d/2$ is for the purpose of accounting for string boundaries). It is not hard to see that the distribution of Z_t does not depend on the value of j.

The total variation distance between the distributions of $r^{(0)}(u)$ and of $r^{(1)}(u)$ equals to the total variation distance between Z_0 and Z_1. We compute the latter via its complement, i.e. the probability mass that is "common" to the two distributions, which is, formally, $\sum_{z\in[-d,d]} \min_{t\in\{0,1\}} \Pr_{r^{(t)}}[Z_t = z]$.

First, we can compute the probability that $Z_t = 0$, i.e., the symbol u_j remains at position j, as follows:

$$\Pr[Z_t = 0] = \frac{d - \mathbb{E}[L]}{d} = 1 - m \cdot \frac{L_{\min}}{\zeta d},$$

irrespective of the value of $t \in \{0, 1\}$.

Next, consider the case when $Z_t \neq 0$ and note that $\Pr_{r^{(0)}}[Z_0 \neq 0] = \Pr_{r^{(1)}}[Z_1 \neq 0] = m \cdot \frac{L_{\min}}{\zeta d}$. We show that, conditioned on $Z_t \neq 0$, the variable Z_t is uniform over most of its support, denoted S_t. Moreover S_0 and S_1 have almost the same size and almost completely overlap. Formally, we prove the following claim.

Claim 7.5.3. *There exists a set $S \subset \mathbb{Z} \setminus \{0\}$ satisfying:*

- *There is $\nu > 0$ such that for each $t \in \{0, 1\}$ and $z \in S$ we have $\Pr_{r^{(t)}}[Z_t = z] = \nu$; and*
- *For each $t \in \{0, 1\}$ we have $\Pr_{r^{(t)}}[Z_t \in S] \geq \frac{m-\xi_1}{m} \cdot \Pr_{r^{(t)}}[Z_t \neq 0]$.*

We first show how Claim 7.5.3 lets us prove that $\delta_1(u) \leq O(\log \alpha \cdot \frac{L_{\min}}{d})$. Indeed, one can observe that $\delta_1(u)$ is bounded by the probability that $\Pr_{r^{(0)}}[Z_0 \notin S \cup \{0\}] = \Pr_{r^{(1)}}[Z_1 \notin S \cup \{0\}]$, which we can bound as

$$\delta_1(u) \leq 1 - \Pr_{r^{(t)}}[Z_t = 0] - \Pr_{r^{(t)}}[Z_t \in S] \leq \frac{\xi_1}{m} \cdot \Pr_{r^{(t)}}[Z_t \neq 0] = O(\log \alpha) \cdot \frac{L_{\min}}{\zeta d}.$$

Proof of Claim 7.5.3. We show the claim for $S = \left\{\pm(1-\epsilon_1)\beta^l L_{\min} \mid l = 1, \ldots, m - \xi_1\right\}$ and $\nu = \frac{1}{2} \cdot \frac{L_{\min}}{\zeta d}$.

Let us consider the case that $Z_t \neq 0$. Then, the magnitude of the displacement, $|Z_t|$, must be either $\epsilon_t L$ or $(1-\epsilon_t)L$ where $L = \beta^l L_{\min}$ for some $l \in [m]$. In particular, $Z_t \neq 0$ iff the position j falls inside the rotation block of the operation $r^{(t)}$, and either: (i) j falls into the bigger part of size $(1-\epsilon_t)L$ (that does not wrap-around), hence $|Z_t| = \epsilon_t L$; or (ii) j falls into the smaller part of size $\epsilon_t L$ (that does wrap-around), hence $|Z_t| = L - \epsilon_t L = (1-\epsilon_t)L$. Moreover, conditioned on the magnitude of Z_t, the sign of Z_t is equi-probable to be either positive or negative (depending on whether the rotation block rotates to the right or left).

When $t = 0$, we can compute the probability that $|Z_0| = \frac{1}{2}L = \frac{1}{2}\beta^l L_{\min}$ for some $l \in [m]$ as follows. We have $Z_0 = L/2$ when we choose block length $L = \beta^l L_{\min}$, which happens with probability β^{-l}/ζ, and additionally either (i) position j is inside the "bigger" part of the block, of size $(1-\epsilon_0)L = L/2$, and the block moves to right, or (ii) position j is inside the "smaller" part of the block, of size $\epsilon_0 L = L/2$, and the block moves to left. Formally,

$$\Pr_{r^{(0)}}[Z_0 = L/2] = \Pr_{r^{(0)}}[Z_0 = -L/2] = \tfrac{\beta^{-l}}{\zeta} \cdot \tfrac{(1-\epsilon_0)\beta^l L_{\min}}{d} \cdot \tfrac{1}{2} + \tfrac{\beta^{-l}}{\zeta} \cdot \tfrac{\epsilon_0 \beta^l L_{\min}}{d} \cdot \tfrac{1}{2} = \tfrac{L_{\min}}{\zeta d} \cdot \tfrac{1}{2} = \nu.$$

Note that $z = \frac{1}{2}\beta^l L_{\min}$ may be written as $z = (1-\epsilon_1)\beta^{l-1} L_{\min}$ (using Eqn. (7.14)) and thus $z \in S$ whenever $l \in \{2, \ldots m - \xi_1 + 1\}$.

Now let $t = 1$. When $|Z_1| = \epsilon_1 \beta^{l+\xi_1} L_{\min} = (1-\epsilon_1) \cdot \beta^l L_{\min} \in S$ for $l \in \{1, \ldots, m-\xi_1\}$ (the equality here is by Eqn. (7.14)), we again have that

$$\Pr_{r^{(1)}}[Z_1 = \epsilon_1 \beta^{l+\xi_1}] = \frac{\beta^{-l-\xi_1}}{\zeta} \cdot \frac{(1-\epsilon_1)\beta^{l+\xi_1} L_{\min}}{d} \cdot \frac{1}{2} + \frac{\beta^{-l}}{\zeta} \cdot \frac{\epsilon_1 \beta^l L_{\min}}{d} \cdot \frac{1}{2} = \frac{L_{\min}}{\zeta d} \cdot \frac{1}{2} = \nu.$$

Finally, note that $\Pr_{r^{(t)}}[Z_t \in S] = \sum_{z \in S} \Pr_{r^{(t)}}[Z_t = z] = 2(m-\xi_1) \cdot \nu = \frac{m-\xi_1}{m} \cdot \Pr_{r^{(t)}}[Z_t \neq 0]$. This concludes the proof of Claim 7.5.3. $\square$

Part 2: $\lambda \geq 2$. When we have $\lambda \geq 2$ non-zero entries in u, the intuition is to group these non-zero entries into one or more "atomic intervals" and then reduce to the case $\lambda = 1$ with the role of "symbol u_j" being replaced by an atomic interval. For example, when there are $\lambda = 2$ non-zero entries in u, most of the block lengths L fall into two categories:

- L is much larger than the distance between the positions of the two non-zero entries — in this case, the two non-zero symbols from u move jointly (atomically) most of the time, and thus the interval connecting the two symbols behaves roughly as the "symbol u_j" in the $\lambda = 1$ scenario;
- L is much smaller than the distance between the two positions — in this case, each of the two non-zero entries can be treated independently as in $\lambda = 1$ case, and we lose only a factor of λ (by "union bound").

Furthermore, we can bound the number of values of L that do not satisfy one of the above properties. A relatively straight-forward bound is $O(\lambda^2)$ (all pair-wise distances between the non-zero entries), times $O(\xi_1)$ (the same extra factor as in the $\lambda = 1$ case). This analysis would give a bound of $\delta_\lambda(u) \leq O(\lambda^3 \log \alpha \cdot \frac{L_{\min}}{d})$. In the sequel we obtain a stronger bound, with only a linear dependence on λ, using a more careful analysis. (For the impact of a weaker bound see the calculation in Section 7.4.2.)

More generally, we partition the non-zero entries of u such that each part consists of "nearby" entries, while the parts are "far" amongst themselves. We then view each part as a contiguous *A-interval* (stands for atomic interval). Once we manage such an approximation, we have several A-intervals (at most λ), and we expect each one to move atomically: all non-zero entries from the same A-interval will move the same direction by the same displacement most of the time. The main challenge lies in the fact that the notion of nearby entries depends on the length L of the rotation block, and we say two non-zero entries are nearby if their positions differ by at most L. Thus, for each possible block length L, we have a possibly different partition of entries into A-intervals (partitions are progressively coarser with bigger L). The main technical work is to analyze the structure of these A-intervals over all lengths L.

We proceed with a complete proof below. For a block length $L = \beta^l L_{\min}$, we define the graph G_L as follows. G_L is an undirected graph on λ vertices, where each vertex corresponds to a non-zero entry in u. For convenience, we use the term "entry" when we refer to the position of a non-zero entry of u, and equivalently a vertex of G_L (in contrast, we will use the term node for another graph structure defined later). We connect two entries $i, j \in [d]$ if $|i-j|^* \leq L$, where $|i-j|^* = \min\{|i-j|, d-|i-j|\}$ computes distance on the d-cycle. For a graph G_L, we focus on its connected components, which may be viewed as intervals in $\mathbb{Z}_d$. Specifically, to each connected component $C \subset V$ we assign the interval $I(C)$, an interval defined as the minimal interval (with wrap-around) on $\mathbb{Z}_d$ that contains all entries in C. Overloading the notation, we write an interval $I(C) = [i, j]$ to mean that $I(C) = \{i, i+1, \dots, j\}$ if $i \le j$ and $I(C) = \{i, i+1, \dots, d, 1, 2, \dots j\}$ if $j < i$. The length of interval $I = [i, j]$ is $\mathrm{len}(I) = |I| = (j - i + 1) (\mathrm{mod}\ d)$. Note that, for every connected component C, every two *consecutive* entries in $I(C)$ are at distance at most L; thus, the length of any interval $I(C)$ can be at most $L \cdot \lambda < d^{0.99}$; also if $I(C) = [i, j]$ then both i and j are non-zero entries of u.

An *A-interval* is then an interval $I(C)$ that corresponds to some connected component C. Each block length L induces potentially different graph G_L, that in turn induces different set of A-intervals. The following observation relates A-intervals induced by different G_L's.

Observation 7.5.4. *If two entries are in the same A-interval (equivalently, connected component) in G_L for some L, then there are also in the same A-interval in $G_{L'}$ for any $L' \ge L$.*

We use this observation to define a forest on all the A-intervals, as follows. The forest consists of m levels, where nodes at level $l \in [m]$ correspond to the A-intervals for $L = \beta^l L_{\min}$ (i.e. the connected components in G_L). For a forest node v at level l we write $I(v)$ for the corresponding A-interval. The edges in the forest are defined as follows: for two forest nodes v_1, v_2 on two consecutive levels, l and $l+1$ respectively, we connect v_1 to v_2 iff $I(v_1) \subseteq I(v_2)$. This construction is well-defined due to Observation 7.5.4. Nodes at level 1 will be called *leaves.* Notice that every forest node at level $l > 1$ indeed has at least one edge to a node at level $l - 1$, i.e. non-leaf nodes have at least one child. Let $n_l \in [\lambda]$ be the number of nodes at level l.

We now wish to bound the error incurred by considering an A-interval to be an atomic object. Specifically, a too long A-interval is likely to move not atomically, in the sense that the interval is "cut" by the rotation block. We bound the error of our "approximation" using the probability that a random position $s \in [d]$ (one of the two block boundaries) falls inside these A-intervals at a random level l. The latter probability is proportional to expected sum of lengths of the A-intervals of G_L, when we choose the block length L randomly according to the distribution $\mathcal{D}_t^{\mathrm{rot}}$.

Claim 7.5.5. *Let $s \in [d]$ be chosen uniformly at random and let $l \in [m]$ be chosen randomly with probability β^{-l}/ζ. Then,*

$$\Pr_{s,l}\Big[s \textit{ is inside one of the A-intervals at level } l\Big] \le O\Big(\lambda \frac{L_{\min}}{d}\Big).$$

Proof. Consider any two *consecutive* non-zero entries of u and let J be the interval between them (with wrap-around), including one of the endpoints, say the left one. We compute next the probability that s is contained in this interval J, and interval J is contained in an A-interval $I(v)$ for a forest node v at level l. Note that summing this probability over all λ intervals J gives the final quantity we want.

By definition, an interval J is inside an A-interval at level l iff $|J| \le \beta^l L_{\min}$. Thus, for a fixed J, the probability that both $s \in J$ and J is contained in an A-interval at level l is at most

$$\frac{|J|}{d} \cdot \sum_{l \in [m]:\ |J| \le \beta^l L_{\min}} \frac{\beta^{-l}}{\zeta} \le \frac{|J|}{d} \cdot \frac{L_{\min}}{\zeta \cdot |J|} \cdot \frac{1}{1 - \beta^{-1}} \le O\left(\frac{L_{\min}}{d}\right).$$

We have exactly λ such intervals J, and thus the total contribution is $O(\lambda \frac{L_{\min}}{d})$. □

We now continue with computing the total variation distance $\delta_\lambda(u)$ between $r^{(0)}(u)$ and $r^{(1)}(u)$ where $r^{(0)} \in \mathcal{D}_0^{\mathrm{rot}}$ and $r^{(1)} \in \mathcal{D}_1^{\mathrm{rot}}$. As in part one ($\lambda = 1$), we will bound the total variation distance between them by estimating the probability mass "common" to the two distributions.

First we compute the probability that all non-zero entries of u stay put (as in part one).

Claim 7.5.6. *For each $t \in \{0, 1\}$, we have that*

$$\Pr_{r^{(t)}}[r^{(t)}(u) = u] \geq 1 - O\left(\lambda \frac{L_{\min}}{d}\right) - \sum_{l=1}^{m} n_l \cdot \frac{L_{\min}}{\zeta d}.$$

Proof. The complement event is that at least one non-zero entry of u is displaced. Whenever it occurs, at least one of the following holds:

- Left or right endpoint of the rotation block belongs to an A-interval induced by G_L; or else
- The rotation block contains inside it an entire A-interval induced by G_L.

The probability of the first event is bounded by, using Claim 7.5.5:

$$2 \Pr_{s,L}[s \text{ is inside one of the A-intervals at level } l] \leq O\left(\lambda \frac{L_{\min}}{d}\right).$$

The probability of the second event can be bounded by the probability that the rotation block includes the leftmost endpoint of some A-interval at level l:

$$\sum_{l=1}^{m} \frac{\beta^{-l}}{\zeta} \sum_{v \text{ at level } l} \Pr_s \Big[\text{left endpoint of } I(v) \text{ is inside } [s, s+L-1]\Big] \leq$$
$$\leq \sum_{l=1}^{m} \frac{\beta^{-l}}{\zeta} \cdot \frac{n_l \cdot L}{d} = \sum_{l=1}^{m} n_l \frac{L_{\min}}{\zeta d}$$

The claim follows from the last two inequalities by applying a union and then considering the complement event. ☐

We now prove a claim that should be seen as the analogue of Claim 7.5.3 from part one, which characterizes the common weight of the distributions of $r^{(0)}(u)$ and $r^{(0)}(u)$ when some entries (more precisely, A-intervals) move. In contrast to part one, here we have to also consider the case when an A-interval does not behave atomically, i.e., when the rotation block intersects the A-interval of some node v at a level $l \in [m]$. This will contribute some additional error term that depends on the length of the interval $I(v)$, and which we will bound using Claim 7.5.5.

Let us define the random variable $Z_t(I)$, for an interval $I = I(v)$ corresponding to a forest node v, and $t \in \{0, 1\}$. $Z_t(I)$ denotes the (position) displacement of the entries from the interval I under rotation $r^{(t)} \in \mathcal{D}_1^{\mathrm{rot}}$ when the interval I moves atomically and no entry outside I moves. We set $Z_t(I) = \perp$ if the interval I does not move atomically and/or some other entry outside I moves as well under $r^{(t)}$.

Claim 7.5.7. *There exists a set $S \subset \mathbb{Z} \setminus \{0\}$ satisfying:*

- *For each interval $I = I(v)$ corresponding to a forest node v at level $l^* \in \{\xi_1+1, \dots m\}$, and for each $t \in \{0,1\}$ and $z \in S$,*

$$\Pr_{r^{(t)}}[Z_t(I) = z] \geq \frac{1}{2} \cdot \frac{L_{\min}}{\zeta d} - \beta^{-(l^*-\xi_1)} \cdot \frac{2\,\mathrm{len}(I)}{\zeta d};$$

- *Call two intervals $I(v)$ and $I(v')$ distinct if they have at least one distinct endpoint; then*

$$\sum_{z\in S} \sum_{\text{distinct } I=I(v)} \min_{t\in\{0,1\}} \left\{ \Pr_{r^{(t)}}[Z_t(I) = z] \right\} \geq \sum_{l=\xi_1+1}^{m} n_l \frac{L_{\min}}{\zeta d} - O\left(\lambda \frac{L_{\min}}{d}\right).$$

Proof. We show the claim for $S = \left\{\pm \left\lfloor (1-\epsilon_1)\beta^l L_{\min} \right\rfloor \mid l = 1, \dots, m-\xi_1 \right\}$.

Fix an interval $I = I(v)$ for a node at level $l^* \geq \xi_1 + 1$. Consider the displacement $z = \left\lfloor \epsilon_1 \beta^{l^*} L_{\min} \right\rfloor = \left\lfloor (1-\epsilon_1)\beta^{l^*-\xi_1} L_{\min} \right\rfloor \in S$ (the equality is again by Eqn. (7.14)). We now bound $\Pr[Z_1(I) = z]$, namely the probability that all the entries in $I(v)$ are moved (atomically) z positions to the right (and all the other entries stay put), under the distribution $\mathcal{D}_1^{\mathrm{rot}}$. We have $\Pr[Z_1(I) = z]$ when either: i) $l = l^*$, interval I is completely inside the "bigger" part of the block, of size $\left\lfloor (1-\epsilon_1)\beta^l L_{\min} \right\rfloor$, and the block moves to right, or ii) $l = l^* - \xi_1$, interval I is completely inside the "smaller" part of the block, of size $\left\lfloor \epsilon_1 \beta^{l-\xi_1} L_{\min} \right\rfloor$, and the block moves to left. Note that in both cases all entries outside I stay put as they are at (position) distance at least $\beta^{l^*} L_{\min} + 1$ from I and thus cannot be inside the rotation block. Formally,

$$\begin{aligned}
\Pr_{r^{(1)}}[Z_1(I) = z] &= \Pr_{r^{(1)}=(\tilde{R}_{s,L})^{\epsilon_1 L},\ L=\beta^l L_{\min}} \left[Z_1(I) = z,\ l = l^*,\ \tilde{R} = \overrightarrow{R} \right] \\
&\quad + \Pr_{r^{(1)}=(\tilde{R}_{s,L})^{\epsilon_1 L},\ L=\beta^l L_{\min}} \left[Z_1(I) = z,\ l = l^* - \xi_1,\ \tilde{R} = \overleftarrow{R} \right] \\
&\geq \frac{\beta^{-l^*}}{\zeta} \cdot \frac{1}{2} \cdot \frac{(1-\epsilon_1)\beta^{l^*} L_{\min} - 1 - \mathrm{len}(I)}{d} \\
&\quad + \frac{\beta^{-(l^*-\xi_1)}}{\zeta} \cdot \frac{1}{2} \cdot \frac{\epsilon_1 \beta^{l^*-\xi_1} L_{\min} - 1 - \mathrm{len}(I)}{d} \\
&\geq \frac{1}{2} \frac{L_{\min}}{\zeta d} - \frac{\beta^{-(l^*-\xi_1)}}{\zeta} \cdot \frac{2\,\mathrm{len}(I)}{d}
\end{aligned} \tag{7.16}$$

Similarly, we can give the exact same lower bound for each of the following four events: $Z_1(I) = \pm z$ and $Z_0(I) = \pm z$.

We can now bound the probability mass that is common to the two distributions $r^{(0)}(u)$ and $r^{(1)}(u)$ for the events that there is a distinct interval I such that $Z_t(I) = z$ for some $z \in S$:

$$\begin{aligned}
&\sum_{z\in S} \sum_{\text{distinct } I=I(v)} \min_{t\in\{0,1\}} \left\{ \Pr_{r^{(t)}}[Z_t(I) = z] \right\} \\
&\qquad \geq \sum_{l=\xi_1+1}^{m} \sum_{v \text{ at level } l} \Pr\left[Z_t(I(v)) \in \left\{ \pm \left\lfloor \epsilon_1 \beta^l L_{\min} \right\rfloor \right\} \right]
\end{aligned} \tag{7.17}$$

because, for each node v at level $l^* \geq \xi_1 + 1$, we can consider the interval $I = I(v)$ and the displacement of $z = z(v) = \lfloor \epsilon_1 \beta^{l^*} L_{\min} \rfloor \in S$. Then all the events $Z_t(I(v)) = \pm z(v)$ are mutually exclusive (over the choice of such v), and hence we obtain the sum from Eqn. (7.17). Furthermore, using Eqn. (7.16), we obtain:

$$
\begin{aligned}
\sum_{z\in S} \sum_{\text{distinct } I=I(v)} & \min_{t\in\{0,1\}} \left\{ \Pr_{r^{(t)}} [Z_t(I) = z] \right\} \\
& \geq \sum_{l^*=\xi_1+1}^{m} 2 \sum_{v \text{ at level } l^*} \left(\frac{1}{2} \cdot \frac{L_{\min}}{\zeta d} - \frac{\beta^{-(l^*-\xi_1)}}{\zeta} \cdot \frac{2\operatorname{len}(I(v))}{d} \right) \\
& \geq \sum_{l^*=\xi_1+1}^{m} n_{l^*} \frac{L_{\min}}{\zeta d} - 4 \sum_{l^*=1}^{m} \sum_{v \text{ at level } l^*} \frac{\beta^{-l^*}}{\zeta} \cdot \frac{\operatorname{len}(I(v))}{d},
\end{aligned}
\tag{7.18}
$$

where we remind that n_l is the number of nodes at level l^*. The last inequality follows from the fact that, for each interval $I = I(v)$ of a node v at level $l^* \geq \xi_1 + 1$, we can charge $\operatorname{len}(I(v))$ to lengths of the intervals of the descendants of v at level $l^* - \xi_1$.

Finally, observe that the last term in Eqn. (7.18), namely $\sum_{l^*} \sum_v \frac{\beta^{-l^*}}{\zeta} \cdot \frac{\operatorname{len}(I(v))}{d}$, is equal precisely to the probability that a random position s falls into an A-interval at level l^*, where l^* is chosen at random according to the distribution $l^* = l$ with probability β^{-l}/ζ. Thus we can use Claim 7.5.5 to bound it from above,

$$
\sum_{l^*=1}^{m} \sum_{v \text{ at level } l^*} \frac{\beta^{-l^*}}{\zeta} \cdot \frac{\operatorname{len}(I(v))}{d} \leq O(\lambda) \cdot \frac{L_{\min}}{\zeta d},
$$

which together with Eqn. (7.18) completes the proof of Claim 7.5.7. □

To summarize, the total probability mass that we accounted to be common for $t = 0$ and $t = 1$ is the sum of (our lower bounds on) the probability that all entries stay put, plus the probability that exactly one distinct interval $I = I(v)$ is displaced by precisely $z \in S$ positions. Combining Claims 7.5.6 and 7.5.7, and using the trivial bound of $n_l \leq \lambda$ for all $l \in [m]$, we obtain:

$$
1-\delta_\lambda(u) \geq 1-O\left(\lambda \cdot \frac{L_{\min}}{d}\right) - \sum_{l=1}^{m} n_l \cdot \frac{L_{\min}}{\zeta d} + \sum_{l=\xi_1+1}^{m} n_l \frac{L_{\min}}{\zeta d} - O\left(\lambda \frac{L_{\min}}{d}\right) \geq 1 - O(\lambda \xi_1) \cdot \frac{L_{\min}}{\zeta d}.
$$

Finally, using the fact that $\xi_1 = O(\log \alpha)$, we conclude Eqn. (7.15), which completes the proof of Lemma 7.5.2. □

7.5.2 Statistical closeness of the distributions $\tilde{\mu}_t$ and μ_t

We prove Lemma 7.4.9 (b) via the following lemma.

Lemma 7.5.8. *For every $t \in \{0, 1\}$, the total variation distance between $\tilde{\mu}_t$ and μ_t is at most $d^{-\Omega(1)}$.*

Proof. First we recall that $\tilde{\mu}_t$ is equal to the distribution μ conditioned on the fact that the generated pair $(x, y) \in \mu$ is legal, i.e., both x and y are permutations and (x, y) are far or close for $t = 0$ or $t = 1$ respectively. Since both x and y are random from $\mathbb{Z}_p^d$, and $p > d^3$, then x and y are both permutations with probability at least $1 - O(1/d)$.

Thus, the total variation distance between $\tilde{\mu}_0$ and μ_0 is at most $\Pr_{\mu_0}[\mathrm{ed}(x,y) \le R] + O(1/d)$. Similarly, total variation distance between $\tilde{\mu}_1$ and μ_1 is at most $\Pr_{\mu_1}[\mathrm{ed}(x,y) > R/\alpha] + O(1/d)$. Thus, it suffices to prove that $\Pr_{\mu_0}[\mathrm{ed}(x,y) \le R] \le d^{-\Omega(1)}$ and $\Pr_{\mu_1}[\mathrm{ed}(x,y) > R/\alpha] \le d^{-\Omega(1)}$. Remember that x is chosen at random, then $z = x + N_\rho$, and y is obtained from z via a sequence of rotation operations.

We choose the constant $C_2 = 20\zeta/\epsilon_0 = 40\zeta$, and condition on the event that x, y, and z are all permutations, which happens with probability $\ge 1 - O(1/d)$. We can describe the distribution μ_t also as follows. Start with a permutation z, and let x be the permutation obtained by modifying every coordinate in z to a new symbol independently with probability $1-\rho$. We may assume, without loss of generality (by renaming symbols), that z is the identity permutation of length d, i.e. for all $i \in [d]$ we have $z(i) = i$, and furthermore with probability ρ we have $x(i) = z(i)$ and $x(i) = i + d$ otherwise. Next, let y be the permutation obtained from z by applying w random rotation operations chosen from $\mathcal{D}_t^{\mathrm{rot}}$.

It will then suffice to prove the following two claims.

Claim 7.5.9. *For both $t \in \{0,1\}$,*

$$\Pr_{\mu_t}\Big[\,\mathrm{ed}(x,z) \le 2\epsilon_1 R\Big] \ge 1 - e^{-d^{\Omega(1)}}.$$

Claim 7.5.10. *For both $t \in \{0,1\}$,*

$$\Pr_{\mu_t}\left[0.1 \le \frac{\mathrm{ed}(z,y)}{R \cdot C_2\epsilon_t/\zeta} \le 10\right] \ge 1 - d^{-\Omega(1)}.$$

We can now obtain the lemma statement from the above two claims, using a union bound, and applying the triangle inequality $|\,\mathrm{ed}(x,y) - \mathrm{ed}(z,y)| \le \mathrm{ed}(x,z)$ (see also Figure 7-2). Indeed, we obtain (i) that for the distribution μ_0, with high probability, $\mathrm{ed}(x,y) \ge (0.1C_2\epsilon_0/\zeta - 2\epsilon_1)R = (2 - 2\epsilon_1)R > R$; and (ii) that for the distribution μ_1, with high probability, $\mathrm{ed}(x,y) \le (10C_2\epsilon_1/\zeta + 2\epsilon_1)R = 402\epsilon_1 R \le \frac{804}{C_1\alpha} \cdot R < R/\alpha$.

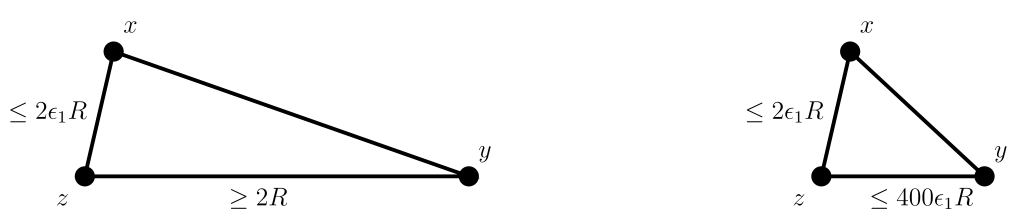


Figure 7-2: The relative positions of x, y, z under the distributions μ_0 and μ_1 respectively.

It remains to prove Claims 7.5.9 and 7.5.10.

Proof of Claim 7.5.9. One can verify that $\mathrm{ed}(x,z)$ is upper bounded by the number of substitutions performed when constructing x from z. This number of substitutions may be bounded using a straightforward Chernoff bound, Fact 2.4.4.

In our case probability of substitution is $q = (1-\rho)(1-1/p)$, where the second factor is the probability that the substituted symbol is different from the original symbol. Since $\rho = 1 - \epsilon_1 R/d$, we get

$$\Pr_{\mu_t}\Big[\,\mathrm{ed}(x,z) \le 2\epsilon_1 R\Big] \ge 1 - e^{-\Omega(\epsilon_1 R)} \ge 1 - e^{-d^{\Omega(1)}}.$$

$\square$

Proof of Claim 7.5.10. We first show an upper bound on $\mathrm{ed}(z, y)$ by analyzing the sum of magnitudes of all the rotation operations. Recall that there are w rotation operations; a single rotation operation works on a block of (random) length $L = \beta^l L_{\min}$, and incurs edit distance at most (in fact, exactly) $2\lfloor \epsilon_t L \rfloor$. For $l \in [m]$, let the random variable Z_l denote the number of rotation operations in which the block length equals $\beta^l L_{\min}$. Observe that Z_l has Binomial distribution $B(w, \frac{\beta^{-l}}{\zeta})$ and its expectation is $\mathbb{E}[Z_l] = w \cdot \frac{\beta^{-l}}{\zeta} \geq \frac{C_2 R}{m L_{\min}} \cdot \frac{d^{-0.01}}{\zeta} \geq d^{\Omega(1)}$. By a straightforward Chernoff bound (Fact 2.4.4),

$$\Pr\left[Z_l \geq 2\mathbb{E}[Z_l]\right] \leq e^{-\Omega(\mathbb{E}[Z_l])} \leq e^{-d^{\Omega(1)}}.$$

Taking a union bound over these events for $l = 1, \ldots, m$, we conclude that with high probability

$$\mathrm{ed}(z, y) \leq \sum_{l=1}^{m} (2w \frac{\beta^{-l}}{\zeta} \cdot 2\epsilon_t \beta^l L_{\min}) = \frac{4C_2 \epsilon_t R}{\zeta}.$$

We proceed to show a lower bound on $\mathrm{ed}(z, y)$, by counting inversions, i.e., pairs of symbols $(a_1, b_1), \ldots, (a_k, b_k)$ such that each a_j appears before b_j in z, but a_j appears after b_j in y. It is easy to verify that if the inversions are disjoint, in the sense that the symbols $a_1, b_1, \ldots, a_k, b_k$ are all distinct, then $\mathrm{ed}(z, y) \geq k$ (because in every alignment of z with y, for each $j = 1, \ldots, k$, at least one of a_j, b_j must incur an edit operation). For each of the w rotation operations we take $\lfloor \epsilon_t L \rfloor$ pairs — simply take the $\lfloor \epsilon_t L \rfloor$ symbols that were at the beginning of the block and match them to the $\lfloor \epsilon_t L \rfloor$ symbols that were at the end of the block. It follows, using Chernoff bounds as above, that with probability at least $1 - e^{-d^{\Omega(1)}}$ this process picks at least $\frac{1}{2} \cdot \frac{C_2 \epsilon_t R}{\zeta}$ pairs of symbols, but this count may include repetitions. Furthermore, a pair "inverted" in one rotation operation may be inverted back by another rotation. To mitigate this concern, fix a pair (a, b) taken at some j-th rotation operation. The probability that symbol a was inside a rotated block in at least one other rotation is at most (using the independence between rotations and a union bound)

$$(w-1) \sum_{l=1}^{m} \left(\frac{\beta^{-l}}{\zeta} \cdot \frac{\beta^l L_{\min}}{d} \right) < \frac{w m L_{\min}}{\zeta d} = \frac{C_2 R}{\zeta d}.$$

A similar argument applies to symbol b, and clearly if both a and b were not inside a rotated block of any of the other $w-1$ rotations, then either (a, b) or (b, a) is an inversion between z and y. It remains to apply a union bound over the $\frac{C_2 \epsilon_t R}{2\zeta}$ pairs of symbols the above process produces, and indeed the probability that at least one of them fails is at most

$$2 \cdot \frac{C_2 \epsilon_t R}{2\zeta} \cdot \frac{C_2 R}{\zeta d} \leq O\left(\frac{R^2}{d}\right) \leq d^{-\Omega(1)}.$$

We conclude that with probability at least $1 - d^{-\Omega(1)}$, the above process produces $\frac{C_2 \epsilon_t R}{2\zeta}$ disjoint inversions, and thus $\mathrm{ed}(y, z) \geq \frac{C_2 \epsilon_t R}{2\zeta}$. This completes the proof of Claim 7.5.10. $\square$

We thus finalized the proof of Lemma 7.5.8. $\square$

7.6 A Poincaré Inequality for Ulam and Edit Distances

Our communication complexity lower bounds (namely Theorem 7.0.6) imply that embedding the edit and Ulam metrics into ℓ_1, and into powers thereof, requires distortion $\Omega(\frac{\log d}{\log\log d})$. But our proof also yields a Poincaré-type inequality, as follows. Indeed, using (i) a variant of Lemma 7.4.4, where $\Pr[\mathcal{H}^A(x) \neq \mathcal{H}^B(x)]$ is replaced with $\mathbb{E}[\mathcal{H}^A(x) - \mathcal{H}^B(x)]^2$ and $\mathcal{H}^A, \mathcal{H}^B$ are real (rather than boolean) functions with $\mathbb{E}_x\left[\mathcal{H}^A(x)\right]^2 = \mathbb{E}_x\left[\mathcal{H}^B(x)\right]^2 = 1$, together with (ii) Lemma 7.5.2 for suitable parameters $R = d^{1/4}$ and $\alpha = \Theta(\frac{\log d}{\log\log d})$, we get that for all $f : \mathbb{Z}_p^d \to \mathbb{R}$ (and thus all $f : \mathbb{Z}_p^d \to \ell_2$)

$$\mathbb{E}_{(x,y)\in\mu_0}\left[f(x) - f(y)\right]^2 - \mathbb{E}_{(x,y)\in\mu_1}\left[f(x) - f(y)\right]^2 \leq \tfrac{1}{10}\mathbb{E}_{x,y\in\mathbb{Z}_p^d}\left[f(x) - f(y)\right]^2. \tag{7.19}$$

In fact, the aforementioned non-embeddability into ℓ_1 (actually into the bigger space squared-ℓ_2) can be proved *directly* from the Poincaré inequality (7.19), as follows. Consider a D-distortion embedding into squared-ℓ_2, namely, let $\phi : \mathbb{Z}_p^d \to \ell_2$ be such that for all permutations $x, y \in \mathbb{Z}_p^d$,

$$\mathrm{ed}(x,y)/D \leq \|\phi(x) - \phi(y)\|_2^2 \leq \mathrm{ed}(x,y)$$

Schoenberg [Sch38] proved (see e.g., [DL97, Theorem 9.1.1]) that for every $\lambda > 0$, applying the transform $x \mapsto 1 - e^{-\lambda x}$ on the distances of a squared-ℓ_2 metric always results with a squared-ℓ_2 metric. Thus, there exists a mapping $\psi : \mathbb{Z}_p^d \to \ell_2$ satisfying

$$\|\psi(x) - \psi(y)\|_2^2 = 1 - e^{-\|\phi(x)-\phi(y)\|_2^2\cdot\alpha/R}.$$

We thus get, using Lemma 7.5.8, that

$$\mathbb{E}_{\mu_0}\|\psi(x) - \psi(y)\|_2^2 - \mathbb{E}_{\mu_1}\|\psi(x) - \psi(y)\|_2^2 - \tfrac{1}{10}\cdot\mathbb{E}_{x,y\in\mathbb{Z}_p^d}\|\psi(x) - \psi(y)\|_2^2 \geq \tfrac{1}{e} - \tfrac{1}{e^{\alpha/D}} - \tfrac{1}{10} - d^{-\Omega(1)}.$$

Combining this inequality with Eqn. (7.19) implies that $D \geq \Omega(\alpha) = \Omega(\frac{\log d}{\log\log d})$.

7.7 Bibliographic Notes

We mention two previous lower bounds on edit distance, which come close to showing a *computational* lower bound for the edit distance.

First, if the operations on the symbols of the strings are restricted to tests of equality, then computing edit distance between two strings over a large alphabet requires $\Omega(d^2)$ comparisons [WC76]. However, this lower bound holds only for exact computation (or $1+o(1)$ approximation) and for strings over a large alphabet (but not for binary strings). In fact, the lower bound breaks down even in the comparison model (when we can compare the relative order of two symbols): e.g., the algorithm of [BFC08] runs in time $O(d^2\frac{(\log\log d)^2}{\log^2 d})$ for computing edit distance between strings over arbitrarily large alphabet.

Second, if we restrict attention to *sublinear*-time, i.e., algorithms that probe only a small part of the two input strings, then there exists a simple separation in terms of query complexity. Specifically, deciding whether the edit distance is $\Omega(d)$ or $O(d^{1-\epsilon})$ requires reading at least $\Omega(d^{1/2-\epsilon/2})$ positions of the strings [BEK+03]. In comparison, the same decision under Hamming distance is achieved easily by sampling $O(1)$ positions. This separation has limited computational implications since it essentially shows that estimating edit distance requires reading many positions of the input strings, and is tied to a particular representation of the inputs.

Chapter 8

Lower Bound for NN under Hamming and Euclidean Distances

In this chapter, we show our lower bound for any data structure solving NN under Hamming and Euclidean distances. In particular, we show that any (randomized) data structure for the $(1+\epsilon)$-approximate near neighbor problem under Hamming or Euclidean distances, which uses a constant number of probes to answer each query, must use $n^{\Omega(1/\epsilon^2)}$ space. This is the first lower bound for this problem in the same model with the upper bound; all previous lower bounds were for models more restrictive than the one in which there exist non-trivial upper bounds. Note that, in the regime of constant number of probes, our space lower bound matches that of the NN algorithms from [KOR00, IM98].

We consider a decision version[1] of the approximate near neighbor problem under the Hamming distance. Given a dataset $D \subset \{0,1\}^d$ of n points and a distance threshold R, build a data structure which, given $q \in \{0,1\}^d$, does the following with probability at least 2/3:

- If there is $p \in D$ such that $\mathrm{H}(q,p) \leq R$, answer YES.
- If there is no $p \in D$ such that $\mathrm{H}(q,p) \leq (1+\epsilon)R$, answer NO.

The lower bound holds for the Euclidean space as well.

As with all known lower bounds for data structures with large space, we consider the *asymmetric communication complexity* of the problem. That is, we consider the setting where two parties, Alice and Bob, are communicating in order to answer a query q. We assume that Alice holds q, while Bob holds D. We show that in order to solve the problem, either Alice sends $\Omega(\frac{1}{\epsilon^2}\lg n)$ bits, or Bob sends $\Omega(n^{1-\delta})$ bits, for any constant $\delta > 0$. By the standard relation to the cell-probe complexity [MNSW98], this implies that the lower bound on space of $2^{\Omega(\epsilon^{-2}\log n)}$. Our result is obtained by showing a close relationship between the complexity of the $(1+\epsilon)$-NN problem and the complexity of the *set disjointness* problem.[2] Lower bounds for the latter problem appeared in [MNSW98] for the case of randomized protocols with one-sided error. We give an analogous (a bit weaker) lower bound for the

[1] The definition of the approximate near neighbor problem employed here is somewhat weaker than the one from the Introduction. Specifically, it does not require the algorithm to provide a "near" point in the YES case. However, this definition is more suitable for the reductions used in this paper. Clearly, the lower bound for this version holds for stronger versions as well.

[2] In the asymmetric context, the set disjointness problem is also called LSD, for Lopsided Set Disjointness.

two-sided error case, solving an open problem posed in that paper. We note the optimal lower bound for the set disjointness problem was recently shown in [Pǎt08].

There has been a considerable number of results on lower bounds for the near and nearest neighbor problems (e.g. see [BOR99, CCGL99, BR02, CR04, PT06] or [Ind03b] for a survey). Most apply to more restrictive (i.e, harder) versions of the problem, where either randomization or approximation are disallowed. For randomized approximation algorithms for the *nearest* neighbor problem, a tight query time bound of $\Theta(\lg \lg d / \lg \lg \lg d)$ is known [CR04], for any constant ϵ and polynomial space.

In contrast to that work, our result holds for the approximate *near* neighbor problem, and establishes a *quantitative* dependence between the approximation factor and the exponent in the space bound (for the constant query time case). Given that the exponent must be quadratic in $1/\epsilon$, our results indicate a fundamental difficulty in designing practical data structures which are very accurate *and* very fast. We note that, very recently, [PTW08] showed another lower bound for the approximate near neighbor problem, for approximations $c > 2$. Their lower bound says that any data structure, with t cell-probes, has to use $n^{1+\Omega(1/c^2/t)}$ space. We point out that their lower bound is *not proven for the decision version* of the NN problem considered in this chapter, but rather for the "full" near neighbor problem, where one has to report an actual near neighbor.

Our space lower bound also holds for a closely related $(1+\epsilon)$-*far* neighbor problem (defined formally in section 8.1.3).

Finally, we mention that our lower bound provides some insight into the broader context of *dimensionality reduction* as used in high-dimensional computational geometry (see also Section 2.3). Specifically, the NN algorithms of [KOR00, IM98] may be seen as an application of the dimensionality reduction method as follows. The idea would be to reduce the dimension of the ambient space to $k = O(\log n/\epsilon^2)$, and then store "all" points in $\mathbb{R}^k$ within distance 1 from each dataset point $p \in D$. To do this, we impose a cubic grid on $\mathbb{R}^k$, with each cell having diameter ϵ. By arguments as in [IM98] it follows that each unit ball in $\mathbb{R}^k$ touches at most $(1/\epsilon)^{O(k)}$ grid cells. Therefore, we can afford to store all such cells within the given space bound. To answer a query, we simply check if the grid cell containing the query point has been stored. Our lower bound suggests that the use of dimensionality-reduction method is essentially tight in the context of the NN problem.

The results from this chapter have previously appeared in [AIP06].

8.1 Lower bounds for the approximate near neighbor

We prove the following data structure lower bound.

Theorem 8.1.1. *Fix $n \in \mathbb{N}$ and $\epsilon \in (0,1)$ such that $\epsilon > n^{-\beta}$ for $\beta < 1/2$. There exists dimension $d = O\left(\frac{\log^2}{\epsilon^5}\right)$ such that the following data structure lower bound holds in the cell-probe model. Suppose the cell size is at most $(d \log n)^{O(1)}$. Then, any data structure for the decision $(1+\epsilon)$-NN problem, with cell-probe complexity of $t \in \mathbb{N}$, must use space $n^{\Omega(1/\epsilon^2/t)}$.*

The result holds for NN under both the Hamming distances in $\{0,1\}^d$ and the Euclidean distance in $\mathbb{R}^d$.

For proving the lower bound, we analyze the asymmetric communication complexity of $(1+\epsilon)$-NN via a reduction from the set disjointness problem. In the set disjointness problem, Alice receives a set S from a universe $[U] = \{1 \dots U\}$, $|S| = m$, and Bob receives a set $T \subset [U]$ of size n. They need to decide whether $T \cap S = \emptyset$. We use the following asymmetric communication complexity lower bound for the latter problem.

Theorem 8.1.2 ([Pǎt08]). *Assume Alice receives a set $S, |S| = m$ and Bob receives a set $T, |T| = n$, both sets coming from a universe of size $2mn$, for $m < n^\gamma$, where $\gamma < 1$ is a constant. In any randomized, two-sided error communication protocol deciding disjointness of S and T, either Alice sends $\Omega(m \lg n)$ bits or Bob sends $\Omega(n^{1-\delta})$ bits, for any $\delta > 0$.*

For completeness, in Section 8.1.2, we present the proof of a slightly weaker version of this theorem, which implies a space exponent of $\frac{1/\epsilon^2}{\log(1/\epsilon)}$ instead of $1/\epsilon^2$.

In Section 8.1.1 we show a reduction from the the set disjointness problem to the NN problem. Thus, using the above theorem for $m = \frac{1}{9\epsilon^2}$, we derive the following theorem on asymmetric communication complexity of the $(1+\epsilon)$-NN problem under the Hamming distance:

Theorem 8.1.3. *Consider the communication complexity version of $(1+\epsilon)$-NN in $\{0,1\}^d$, $d = O(\frac{\log^2 n}{\epsilon^5})$, where Alice receives the query $q \in \{0,1\}^d$ and Bob receives the set $D \subset \{0,1\}^d$. Then, for any $\epsilon = \Omega(n^{-\gamma})$, $\gamma < 1/2$, in any randomized protocol deciding the $(1+\epsilon)$-NN problem, either Alice sends $\Omega(\frac{\log n}{\epsilon^2})$ bits or Bob sends $\Omega(n^{1-\delta})$ bits, for any $\delta > 0$.*

The lower bound on NN under the Euclidean distance follows immediately from the fact that NN under the Hamming distance, for threshold $R > 0$ and approximation $1+\epsilon$ reduces immediately to NN under the Euclidean distance, for threshold $\sqrt{R}$ and approximation $\sqrt{1+\epsilon}$.

The main theorem, Theorem 8.1.1, then follows from Theorem 8.1.3. Specifically, we apply Lemma 1 from [MNSW98], which states:

Lemma 8.1.4 ([MNSW98], Lemma 1). *If there is a solution to the data structure problem with space s, query time t, and cell size b, then there exists a protocol where Alice sends $2t\lceil \log s \rceil$ bits and Bob sends $2tb$ bits.*

For $t = O(1)$, and cell size $b < O(n^{1-\delta})$, for some $\delta > 0$, Bob sends an insufficient number of bits. Thus, Alice needs to send $2t\lceil \log s \rceil > \Omega(m \log n)$ bits. Solving for s, we obtain that space is $s = n^{\Omega(1/\epsilon^2)}$. Note that the cell size b is usually much smaller than $n^{1-\delta}$, typically $b = d \log^{O(1)} n$.

8.1.1 Reduction from asymmetric set disjointness to $(1+\epsilon)$-NN

We prove that we can reduce asymmetric set disjointness problem to the approximate near neighbor. A randomized $[a,b]$-protocol for a communication problem is a protocol in which Alice sends a bits and Bob sends b bits, and the error probability of the protocol is bounded away from $1/2$.

Lemma 8.1.5. *Suppose there exists a randomized $[a,b]$-protocol for the $(1+\epsilon)$-NN problem with $d = O\left(\frac{\log^2 n}{\epsilon^5}\right)$, where Alice receives the query $q \in \{0,1\}^d$ and Bob receives the dataset $D \subset \{0,1\}^d$ of size n. Then there exists a randomized $[a,b]$-protocol for asymmetric set disjointness in an arbitrary universe $[U]$, where Alice receives a set $S \subset [U]$ of size $m = \frac{1}{9\epsilon^2}$, and Bob receives a set $T \subset U$ of size n.*

Proof. We show how to map an instance of asymmetric set disjointness, given by T and S, into an instance of $(1+\epsilon)$-NN, given by respectively the dataset $D \subset \{0,1\}^d$ and the query $q \in \{0,1\}^d$. For this purpose, first, Alice and Bob map their sets S and T into query $\tilde{q} \in \mathbb{R}^U$ and dataset $\tilde{D} \subset \mathbb{R}^U$, i.e., an $(1+\epsilon)$-NN instance in Euclidean U-dimensional space, ℓ_2^U. Then, Alice and Bob map their points from the ℓ_2^U metric to Hamming cube $\{0,1\}^{O(\epsilon^{-5}\log^2 n)}$, essentially preserving the distances among all the points $\tilde{q}$ and $\tilde{D}$. This method for reducing a communication complexity problem into an approximate problem involving Hamming distance appeared earlier in [IW03], albeit in the context of different problems.

For the set $T \subset [U]$, we define $\tilde{D} \triangleq \{e_u \mid u \in T\}$, where e_u is a standard $\mathbb{R}^d$ basis vector, with 1 in the u^{th} coordinate, and 0 everywhere else. For the set S, we set the query $\tilde{q} \triangleq 3\epsilon \cdot \sum_{u\in S} e_u$; note that $\|\tilde{q}\|_2^2 = m \cdot (3\epsilon)^2 = 1$.

We show that if $S \cap T = \emptyset$, then $\|\tilde{q} - \tilde{p}\|_2 = \sqrt{2}$ for all $\tilde{p} \in \tilde{D}$, and, if $S \cap T \neq \emptyset$, then there exists a point $\tilde{p} \in \tilde{D}$ such that $\|\tilde{q} - \tilde{p}\|_2 \leq (1 - \frac{4\epsilon}{3})\sqrt{2}$. Indeed, we have that

- if $S \cap T = \emptyset$, then for any $\tilde{p} \in \tilde{D}$, we have that $\|\tilde{q} - \tilde{p}\|_2^2 = \|\tilde{q}\|_2^2 + \|\tilde{p}\|_2^2 - 2\tilde{q} \cdot \tilde{p} = 2$;
- if $S \cap T \neq \emptyset$, then for $u^* \in S \cap T$ and for $\tilde{p} = e_{u^*} \in \tilde{D}$, we have $\|\tilde{q} - \tilde{p}\|_2^2 = \|\tilde{q}\|_2^2 + \|\tilde{p}\|_2^2 - 2\tilde{q} \cdot \tilde{p} = 2 - 2(3\epsilon e_{u^*}) \cdot e_{u^*} = 2(1 - 3\epsilon)$.

To construct $D \subset \{0,1\}^d$ and $q \in \{0,1\}^d$, Alice and Bob perform a randomized mapping of ℓ_2^U into $\{0,1\}^d$ for $d = O(\epsilon^{-5}\log^2 n)$, such that the distances are only insignificantly distorted, with high probability. Alice and Bob use a source of public random coins to construct the same randomized mapping. First, they construct a randomized embedding f_1 mapping ℓ_2^U into $\ell_1^{O(\epsilon^{-2}\log n)}$ with distortion less than $(1+\epsilon/16)$ (see, e.g., [Ind01a]). Then, they construct the standard embedding f_2 mapping $\ell_1^{O(\epsilon^{-2}\log n)}$ into $\{0,1\}^{O(\epsilon^{-5}\log^2 n)}$. The embedding f_2 first scales up all coordinates by $Z = O(\frac{\log n}{\epsilon^3})$, then rounds the coordinates, and finally transforms each coordinate into its unary representation. We set the constants such that the resulting approximation of f_2 is an additive term $O(\frac{\log n}{\epsilon^2}) < \frac{Z\epsilon\sqrt{2}}{16}$.

Next, Alice and Bob construct $q = f_2(f_1(\tilde{q})) \in \{0,1\}^d$ and $D = \{f_2(f_1(\tilde{p})) \mid \tilde{p} \in \tilde{D}\} \subset \{0,1\}^d$. Notice that for any $p = f_2(f_1(\tilde{p})) \in D$, if $\|\tilde{q} - \tilde{p}\|_2 \geq \sqrt{2}$, then $\|q - p\|_H \geq Z\sqrt{2}(1 - \epsilon/16) - \frac{Z\epsilon\sqrt{2}}{16} = Z\sqrt{2}(1 - \frac{\epsilon}{8})$, and if $\|\tilde{q} - \tilde{p}\|_2 \leq \sqrt{2}(1 - \frac{4\epsilon}{3})$, then $\|q - p\|_H \leq Z\sqrt{2}(1 - \frac{4\epsilon}{3})(1+\epsilon/16) + \frac{Z\epsilon\sqrt{2}}{16} \leq Z\sqrt{2}(1 - \epsilon - \frac{5\epsilon}{24})$.

Finally, Alice and Bob can run the $(1+\epsilon)$-NN communication protocol with $R = Z\sqrt{2}(1 - \epsilon - \frac{5\epsilon}{24})$ to decide whether $S \cap T = \emptyset$. Note that the error probability of the resulting set disjointness protocol is bounded away from 1/2 since $(1+\epsilon)$-NN communication protocol has error probability bounded away from 1/2, and the embedding $f_2 \circ f_1$ fails with probability at most $n^{-\Omega(1)}$. □

8.1.2 A lower bound for asymmetric set disjointness

In this section, we prove a slightly weaker version of Theorem 8.1.2:

Theorem 8.1.6. *Assume Alice receives a set $S, |S| = m$ and Bob receives a set $T, |T| = n$, both sets coming from a universe of size $2mn$, for $m < n^\gamma$, where $\gamma < 1/3$ is a constant. In any randomized, two-sided error communication protocol deciding disjointness of S and T, either Alice sends $\Omega(\frac{m}{\log m}\lg n)$ bits or Bob sends $\Omega(n^{1-\delta}/m^2)$ bits, for any $\delta > 0$.*

First we define the hard instance. The elements of our sets come from the universe $[2m] \times [n]$. Alice receives $S = \{(i, s_i) \mid i \in [m]\}$, for $s_1, \dots, s_m$ chosen independently at random from $[n]$. Bob receives $T = \{(t_j, j) \mid j \in [n]$, for $t_1, \dots, t_n$ chosen independently from $[2m]$. The output should be 1 iff the sets are disjoint. Note that the number of choices is n^m for S and $(2m)^n$ for T, and that S and T are chosen independently.

The lower bound follows from the following variant of the richness lemma, based on [MNSW98, Lemma 6]. The only change is that we make the dependence on ϵ explicit, because we will use $\epsilon = o(1)$.

Lemma 8.1.7. *Consider a problem $f : X \times Y \to \{0, 1\}$, and some probability distributions η_X, η_Y over sets X, Y respectively. Suppose $\Pr_{x \in X, y \in Y}[f(x, y) = 0] \geq \Omega(1)$.*

If f has a randomized two-sided error $[a, b]$-protocol, then there is a rectangle $\mathcal{X} \times \mathcal{Y}$ of f of sizes at least $\eta_X(\mathcal{X}) \geq 2^{-O(a \log 1/\epsilon)}$ and $\eta_Y(\mathcal{Y}) \geq 2^{-O((a+b) \log 1/\epsilon)}$ in which the density (i.e., conditional measure) of ones is at most ϵ. Also, the protocol outputs value 0 on $\mathcal{X} \times \mathcal{Y}$.

To apply the lemma, we first show the disjointness function is 1 with constant probability.

Lemma 8.1.8. *As S and T are chosen randomly as described above,* $\Pr[S \cap T = \emptyset] = \Omega(1)$.

Proof. Note that $S \cap T \subset [n] \times [m]$. We have $\Pr[(i, j) \in S \cap T] = \frac{1}{n(2m)}$ when $i \in [n], j \in [m]$. Then by linearity of expectation $\mathbb{E}_[[|]S \cap T|] = \frac{1}{2}$. Since $|S \cap T| \in \{0, 1, 2, \dots\}$, we must have $\Pr[|S \cap T| = 0] \geq \frac{1}{2}$. □

Thus, it remains to show that no big enough rectangle has a small density of zeros. Specifically, we show the following:

Lemma 8.1.9. *Let $\delta > 0$ be arbitrary. If we choose $S \in \mathcal{S}, T \in \mathcal{T}$ uniformly and independently at random, where $|\mathcal{S}| > 2n^{(1-\delta)m}$ and $\mathcal{T} \geq (2m)^n \cdot 2/e^{n^{1-\delta}/(8m^2)}$, then the probability $S \cap T \neq \emptyset$ is at least $\frac{1}{16m^2}$.*

We use the richness lemma with $\epsilon = \frac{1}{32m^2}$. If there exists an $[a, b]$ protocol for our problem, we can find a rectangle of size $\left(n^m/2^{O(a \lg m)}\right) \times \left((2m)^n/2^{O((a+b) \lg m)}\right)$, in which the fraction of zeros is at most ϵ. To avoid contradicting Lemma 8.1.9, we must either have $2^{O(a \lg m)} > n^{\delta m}/2$, or $2^{O((a+b) \lg m)} > e^{n^{1-\delta}/(8m^2)}/2$. This means either $a = \Omega(\frac{m}{\lg m} \lg n)$ or $a + b = \Omega(n^{1-\delta}/(m^2 \lg m))$. If $m < n^\gamma$, for constant $\gamma < \frac{1}{3}$, this implies that $a = \Omega(\frac{m}{\lg m} \lg n)$ or $b = \Omega(n^{1-\delta}/m^2)$, for any $\delta > 0$.

Proof. (of Lemma 8.1.9) Choosing S at random from $\mathcal{S}$ induces a marginal distribution on $[n]$. Now consider the heaviest $n^{1-\delta}$ elements in this distribution. If the total probability mass of these elements is at most $1 - \frac{1}{2m}$, we call i a *well-spread coordinate.*

Lemma 8.1.10. *If $|\mathcal{S}| > 2n^{(1-\delta)m}$, there exists a well-spread coordinate.*

Proof. Assume for contradiction that no coordinate is well-spread. Consider the set $\mathcal{S}'$ formed by $S \in \mathcal{S}$ such that no s_i is outside the heaviest $n^{1-\delta}$ elements in S_i. By a union bound, the probability over $S \in \mathcal{S}$ that some s_i is not among the heavy elements is at most $m\frac{1}{2m} = \frac{1}{2}$. Then, $|\mathcal{S}'| \geq |\mathcal{S}|/2$. On the other hand $|\mathcal{S}'| \leq (n^{1-\delta})^m$, since for each coordinate we have at most $n^{1-\delta}$ choices. This contradicts the lower bound on $|\mathcal{S}|$. □

Let i be a well-spread coordinate. We now lower bound the probability of $S \cap T \neq \emptyset$ by the probability of $S \cap T$ containing an element on coordinate i. Furthermore, we ignore the $n^{1-\delta}$ heaviest elements of S_i. Let the remaining elements be W, and $p(j) = \Pr[s_i = j]$ when $j \in W$. Note that $p(j) \leq 1/n^{1-\delta}$, and $\sum_{j \in W} p(j) \geq \frac{1}{2m}$.

Define $\sigma(T) = \sum_{j \in W: t_j = i} p(j)$. For some choice of T, $\sigma(T)$ gives exactly the probability of an interesting intersection, over the choice of $S \in \mathcal{S}$. Thus, we want to lower bound $\mathbb{E}_T [\sigma(T) \mid T \in \mathcal{T}]$.

Assume for now that T is uniformly distributed in the original space (not in the subspace $\mathcal{T}$). Note that $\sigma(T) = \sum_{j \in W} X_j$, where X_j is a variable equal to $p(j)$ when $t_j = i$ and 0 otherwise. By linearity of expectation, $\mathbb{E}_T [\sigma(T)] = \sum_{j \in W} \frac{p(j)}{2m} \geq 1/(2m)^2$. Since X_j's are independent (t_j's are independent when T is not restricted), we can use a Chernoff bound to deduce $\sigma(T)$ is close to this expectation with very high probability over the choice of T. Indeed, $\Pr[\sigma(T) < \frac{1}{2} \cdot \frac{1}{(2m)^2}] < e^{-n^{1-\delta}/(8m^2)}$.

Now we can restrict ourselves to $T \in \mathcal{T}$. The probability $\sigma(T) < \frac{1}{8m^2}$ is so small, that it remains small even in this restricted subspace. Specifically, this probability is at most $\Pr[\sigma(T) < \frac{1}{8m^2}] / \Pr[T \in \mathcal{T}] \leq \exp\left[-n^{1-\delta}/(8m^2)\right] / (2 \exp\left[-n^{1-\delta}/(8m^2)\right]) = \frac{1}{2}$. Since $\sigma(T) \geq 0, (\forall) T$, we conclude that $\mathbb{E}_T [\sigma(T) \mid T \in \mathcal{T}] \geq \frac{1}{2} \cdot \frac{1}{8m^2} = \frac{1}{16m^2}$. □

8.1.3 Approximate far neighbor problem

The above lower bound for the $(1+\epsilon)$-NN problem can also be transferred to the $(1+\epsilon)$-*far* neighbor problem, yielding exactly the same space lower bound. Formally, we define the $(1+\epsilon)$-far neighbor as follows. Given a set $D \subset \{0,1\}^d$ of n points and a distance R, build a data structure which given $q \in \{0,1\}^d$ does the following, with probability at least, say, $2/3$:

- If there is $p \in D$ such that $\|q - p\| \geq R$, answer YES.
- If there is no $p \in D$ such that $\|q - p\| \geq R/(1+\epsilon)$, answer NO.

The lower bound results from the following lemma, an equivalent of Lemma 8.1.5.

Lemma 8.1.11. *Suppose there exists a randomized $[a, b]$-protocol for the $(1+\epsilon)$-far neighbor problem with $d = O\left(\frac{\log^2 n}{\epsilon^5}\right)$, where Alice receives the query $q \in \{0,1\}^d$ and Bob receives the dataset $D \subset \{0,1\}^d$ of size n. Then there exists a randomized $[a, b]$-protocol for asymmetric set disjointness in an arbitrary universe $[U]$, where Alice receives a set $S \subset [U]$ of size $m = \frac{1}{9\epsilon^2}$, and Bob receives a set $T \subset U$ of size n.*

As before, together with theorem 8.1.2, this lemma implies that any data structure for $(1+\epsilon)$-far neighbor problem achieving constant number of cell probes, has space $n^{\Omega(1/\epsilon^2)}$.

Proof of Lemma 8.1.11. Same as the proof of lemma 8.1.5, except set the query $\tilde{q} = -3\epsilon \sum_{u \in S} e_u$. □

Chapter 9

Lower Bound for NN under ℓ_∞

In this chapter, we present our lower bound for the NN under the ℓ_∞ distance. We show that, for any $\rho > 1$, any deterministic decision tree for the $O(\log_\rho \log d)$-NN under the d-dimensional ℓ_∞ distance, of strongly sub-linear depth, must have $n^{\Omega(\rho)}$ size. In particular, one obtains an (unusual) double-logarithmic approximation for polynomial space. This space-approximation trade-off matches that of the algorithm given in [Ind98]. The latter paper constructs an NN algorithm for d-dimensional ℓ_∞ with approximation $4\lceil \log_\rho \log 4d \rceil + 1$, which requires space $dn^\rho \log^{O(1)} n$ and $O(d \cdot \log n)$ query time, for any $\rho > 1$. Furthermore, the data structure from [Ind98] is a deterministic decision tree, of depth $O(d \log n)$. For 3-approximation, [Ind98] also gives a $n^{\log d + 1}$ space algorithm, with a similar query time.

Our lower bound thus gives evidence for the apparent oddity of the ℓ_∞ norm. Specifically, while ℓ_∞ is one of the classical ℓ_p norms, it seems to present a very different behavior from, say, Hamming (ℓ_1) and Euclidean (ℓ_2) norms. The NN algorithms under Hamming (ℓ_1) and Euclidean (ℓ_2) distances achieve much better approximations (constant), but the data structures are randomized (in fact, in the case of these two spaces, randomization seems to be required). Furthermore, essentially the only previous worst-case result on ℓ_∞ is the aforementioned algorithm of [Ind98], leaving ℓ_∞ norm quite poorly understood.

These considerations have lead to the optimism that one can achieve NN under ℓ_∞ with a constant factor approximation in polynomial space [Ind01b]. Our lower bound shows one cannot achieve this goal with (deterministic) decision trees, and thus make the goal somewhat elusive.

There are multiple motivations for studying the ℓ_∞ distance, both intrinsic and extrinsic. First, by itself, ℓ_∞ may be a natural choice as a similarity metric for some applications, especially when coordinates are rather heterogeneous. If the features represented by coordinates are hard to relate, it is hard to add up their differences numerically, in the sense of the ℓ_1 or ℓ_2 norm (the "comparing apples to oranges" phenomenon). One popular proposal is to convert each coordinate to rank space, and use the maximum rank difference as an indication of similarity. See for example [Fag96, Fag98].

Besides being a natural distance, ℓ_∞ is a promising host for embedding other metrics into it. For example, a classical result of Matoušek states that *any* metric on n points can be embedded into ℓ_∞ with dimension $d = O(cn^{1/c} \log n)$ and distortion $2c - 1$ [Mat96]. While the dimension of $n^{\Theta(1)}$ is too high for most applications (say, for the edit distance on $\{0,1\}^d$, we have $n = 2^d$), this embedding illustrates some of the power of ℓ_∞.

Early embeddings into ℓ_∞ with interesting dimension included various results for Hausdorff metrics [FCI99], embedding tree metrics into dimension $O(\log n)$ [LLR94], and planar graphs metrics into dimension $O(\log n)$ [KLMN04] (improving over [Rao99]).

Even more relevant to our discussion is the fact that, as shown in Chapter 5, ℓ_∞ (of low dimension) plays a fundamental role in the design of our new NN algorithms for the Ulam and EMD distances, as well as, more generally, for the product spaces and the ℓ_p spaces for $p > 2$. In fact, our algorithms for all these spaces rely crucially on the algorithm from [Ind98] and the algorithm for NN under max-product from [Ind02b] (which extends [Ind98]). The (nearly) double-logarithmic bounds that we obtain in Chapter 5 come precisely from the double-logarithmic approximation of the algorithms from [Ind98, Ind02b].

Thus, the bottleneck in the best–known algorithms for the Ulam, and ℓ_p for $p > 2$ distances is the ℓ_∞ distance. Our lower bound is a very timely indication that, if further improvement for these metrics is possible, it has to avoid ℓ_∞ and max-product spaces. (We note that NN under max-product is a generalization of NN under ℓ_∞, and thus lower bounds on NN under ℓ_∞ carry over immediately to NN under max-products.)

Previously, the only lower bound for NN under ℓ_∞ was a simple reduction of [Ind98] that shows ℓ_∞ with approximation better than 3 is as hard as the partial match problem; see [JKKR04, Pătt08] for partial match lower bounds.

Statement of the result. As in the previous chapter, we approach the problem via *asymmetric communication complexity*. We consider a setting where Alice holds the query point and Bob holds the set D on n database points. For convenience, we assume that the space is discretized. As in Chapter 8, we study the *decision* version of the approximate near neighbor problem. Specifically, the communication complexity problem is defined as follows. Alice is given a "query point" $q \in \{-m, \dots m\}^d$, and Bob is given the "dataset" $D \subset \{-m, \dots m\}^d$ of size n. Then the c-NN problem is a promise problem in which the two players must:

- output 1 if there exists some $p \in D$ such that $\|q - p\|_\infty \le 1$;
- output 0 if, for all $p \in D$, we have that $\|q - p\|_\infty > c$.

We show the following lower bound on the communication complexity of this problem, which is asymptotically optimal by the algorithm from [Ind98].

Theorem 9.0.12. *Fix $\delta, \epsilon > 0$. Consider a dimension d satisfying $\Omega(\log^{1+\epsilon} n) \le d \le o(n)$, and an approximation ratio c satisfying $3 < c \le O(\log\log d)$. Further define $\rho = \frac{1}{2}(\frac{\epsilon}{4}\log d)^{1/c} > 10$.*

In a deterministic protocol solving c-NN, either Alice sends $a = \Omega(\delta\rho\log n)$ bits, or Bob sends $b = \Omega(n^{1-\delta})$.

Data structures. Asymmetric communication lower bounds imply cell-probe lower bounds for data structures by constructing the natural communication protocol in which Alice sends a cell address in each round, and Bob replies with the cell contents. Thus by a standard analysis of [MNSW98], our communication lower bound implies:

Corollary 9.0.13. *Consider any cell-probe data structure solving d-dimensional near-neighbor search under ℓ_∞ with approximation $c = O(\log_\rho \log d)$. If the word size is $w = n^{1-\delta}$ for some $\delta > 0$, the data structure requires space $n^{\Omega(\rho/t)}$ for cell-probe complexity t.*

As with all large-space lower bounds known to date, this bound is primarily interesting for constant query time, and degrades exponentially with t. We expect this dependence on

t to be far from optimal, but proving a tight lower bound for superconstant t is well beyond the reach of current techniques.

By another standard reduction to the decision tree model (see [KN97] and Section 9.5), we have the following:

Corollary 9.0.14. *Let $\delta > 0$ be arbitrary. A decision tree of depth $n^{1-2\delta}$ and node size n^{δ} that solves d-dimensional near-neighbor search under ℓ_∞ with approximation $c = O(\log_\rho \log d)$, must have size $n^{\Omega(\rho)}$.*

Unlike cell-probe complexity, where the bound degrades quickly with the query time, the lower bound for decision trees holds even for extremely high running time (depth) of $n^{1-\delta}$. A decision tree with depth n and predicate size $O(d \log M)$ is trivial: simply test all database points.

The algorithm from [Ind98] is a decision tree with depth $d \cdot \log^{O(1)} n$ and predicate size $O(\log(n + M))$, which achieves the same interesting trade-off between approximation and space. Thus, we show that this trade-off is optimal, at least in the decision tree model. In particular, for polynomial space, the approximation factor of $\Theta(\lg \lg d)$ is intrinsic to NN under ℓ_∞.

Technical discussion. Perhaps the most innovative component of our lower bound is the conceptual step of understanding why this dependence on the approximation "should" be optimal. In Section 9.1, we recast the idea behind the algorithm from [Ind98] in an information-theoretic framework that explains the behavior of the algorithm more clearly.

This understanding suggests a heavily biased distribution over the database points, which elicits the worst behavior. On each coordinate, the probability of some value x decays doubly-exponentially with x, more precisely as $2^{-(2\rho)^x}$. All d dimensions are independent and identically distributed.

By a standard analysis in communication complexity, Alice's communication will fix the query to be in a set S whose measure in our probability space is bounded from below. Technically, the crucial step is to determine the probability that some point in the database lies in the neighborhood of S. The neighborhood $N(S)$ is the Minkowski sum of the set with the ℓ_∞ ball $[-1, 1]^d$. In other words, $N(S)$ is the set of points that *could* be a nearest neighbor. To find an instance for which the algorithm makes a mistake, we must prove a lower bound on the measure of the neighborhood $N(S)$, showing that a point will fall in the neighborhood with good probability.

The crux of the lower bound is not in the analysis of the communication protocol (which is standard), but in proving a lower bound for $N(S)$, i.e. in proving an isoperimetric inequality. Of course, the initial conceptual step of defining an appropriate biased distribution was the key to obtaining the isopermetric inequality that we need. The proof is rather non-standard for an isoperimetric inequality, because we are dealing with a very particular measure on a very particular space. Fortunately, a few mathematical tricks save the proof from being too technical.

The communication complexity steps are described in Section 9.2. The isoperimetric inequality is shown in Section 9.3.

Randomized lower bounds. As explained above, our lower bound uses distributions on the input rather pervasively, but still, it only works for deterministic protocols. (Fortunately, the upper bound is also deterministic...)

It would be a nice technical development to also show this lower bound for a bounded-error protocol. Unfortunately, this seems beyond the scope of existing techniques. The trouble is that all analyzes of asymmetric communication games have been unable to employ non-product distributions.

In Section 9.4, we show the following interesting factlet: it is not possible to prove asymmetric communication lower bounds over product distributions, for the NN problem with approximation $c > 3$. Thus, a randomized lower bound would need to develop new tools in communication complexity.

The results from this chapter have previously appeared in [ACP08].

9.1 Review of the Upper Bound

Decision trees. Due to the decomposability of ℓ_∞ as a maximum over coordinates, a natural idea is to solve NN by a decision tree in which every node is a coordinate comparison. A node v is reached for some set $Q_v \subseteq \{-m, \ldots, +m\}^d$ of queries. If the node compares coordinate $i \in [d]$ with a "separator" x, its two children will be reached for queries in $Q_\ell = Q_v \cap \{q \mid q_i < x\}$, respectively in $Q_r = Q_v \cap \{q \mid q_i > x\}$ (assume x is non-integral to avoid ties).

Define $[x, y]_i = \{p \mid p_i \in [x, y]\}$. Then, $Q_\ell = Q_v \cap [-\infty, x]_i$ and $Q_r = Q_v \cap [x, \infty]_i$.

If the query is known to lie in some Q_v, the set of database points that could still be a near neighbor is $N_v = D \cap \big(Q_v + [-1, 1]^d\big)$, i.e. the points inside the Minkowski sum of the query set with the ℓ_∞ "ball" of radius one. For our example node comparing coordinate $i \in [d]$ with x, the children nodes have $N_\ell = N_v \cap [-\infty, x+1]_i$, respectively $N_r = N_v \cap [x-1, +\infty]_i$.

Observe that $N_\ell \cap N_r = N_v \cap [x-1, x+1]_i$. In some sense, the database points in this slab are being "replicated," since both the left and right subtrees must consider them as potential near neighbors. This recursive replication of database points is the cause of superlinear space. The contribution of [Ind98] is an intriguing scheme for choosing a separator that guarantees a good bound on this recursive growth.

Information progress. Our first goal is to get a handle on the growth of the decision tree, as database points are replicated recursively. Imagine, for now, that queries come from some distribution μ. The reader who enjoys worst-case algorithms need not worry: μ is just an analysis gimmick, and the algorithm will be deterministic.

We can easily bound the tree size in terms of the measure of the smallest Q_v ever reached: there can be at most $1/\min_v \Pr_\mu[Q_v]$ distinct leaves in the decision tree, since different leaves are reached for disjoint Q_v's. Let $I_Q(v) = \log_2 \frac{1}{\Pr_\mu[Q_v]}$; this can be understood as the information learned about the query, when computation reaches node v. We can now rewrite the space bound as $O\big(2^{\max_v I_Q(v)}\big)$.

Another quantity that can track the behavior of the decision tree is $H_N(v) = \log_2 |N_v|$. Essentially, this is the "entropy" of the identity of the near neighbor, assuming that one exists.

At the root λ, we have $I_Q(\lambda) = 0$ and $H_N(\lambda) = \lg n$. Decision nodes must reduce the entropy of the near neighbor until H_N reaches zero ($|N_v| = 1$). Then, the algorithm can simply read the single remaining candidate, and test whether it is a near neighbor of the query. Unfortunately, decision nodes also increase I_Q along the way, increasing the space bound. The key to the algorithm is to balance this tension between reducing the entropy of the answer, H_D, and not increasing the information about the query, I_Q, too much.

In this information-theoretic view, algorithm of [Ind98] shows that we can (essentially) always find a separator that decreases H_N by some δ but does not increase I_Q by more than $\rho \cdot \delta$. Thus, H_D can be pushed from $\lg n$ down to 0, without ever increasing I_Q by more than $\rho \lg n$. That is, space $O(n^\rho)$ is achieved.

Searching for separators. At the root λ, we let $i \in [d]$ be an arbitrary coordinate, and search for a good separator x on that coordinate. Let π be the frequency distribution (the empirical probability distribution) of the projection on coordinate i of all points in the database. To simplify expressions, let $\pi(x : y) = \sum_{j=x}^{y} \pi(j)$.

If x is chosen as a separator at the root, the entropy of the near neighbor in the two child nodes is reduced by:

$$\begin{aligned} H_N(\lambda) - H_N(\ell) &= \log_2 \tfrac{|N_\lambda|}{|N_\ell|} \\ &= \log_2 \tfrac{|D|}{|D \cap [-\infty, x+1]_i|} = \log_2 \tfrac{1}{\pi(-\infty : x+1)} \\ H_N(\lambda) - H_N(r) &= \log_2 \tfrac{1}{\pi(x-1 : \infty)} \end{aligned}$$

Remember that we have not yet defined μ, the assumed probability distribution on the query. From the point of view of the root, it only matters what probability μ assigns to Q_ℓ and Q_r. Let us reason, heuristically, about what assignments are needed for these probabilities in order to generate difficult problem instances. If we understand the most difficult instance, we can use that setting of probabilities to obtain an upper bound for all instances.

First, it seems that in a hard instance, the query needs to be close to some database point (at least with decent probability). In our search for a worst case, let's just assume that the query is always planted in the neighborhood of a database point; the problem remains to find this near neighbor.

Assume by symmetry that $H_N(\ell) \geq H_N(r)$, i.e. the right side is smaller. Under our heuristic assumption that the query is planted next to a random database point, we can lower bound $\Pr_\mu[Q_r] \geq \pi(x+1, \infty)$. Indeed, whenever the query is planted next to a point in $[x+1, \infty]_i$, it cannot escape from $Q_r = [x, \infty]_i$. Remember that our space guarantee blows up when the information about Q_v increases quickly (i.e. the probability of Q_v decreases). Thus, the worst case seems to be when $\Pr_\mu[Q_r]$ is as low as possible, namely equal to the lower bound.

Thus, we have convinced ourselves that it's reasonable to define μ such that:

$$\Pr_\mu[Q_\ell] = \pi(-\infty : x+1); \qquad \Pr_\mu[Q_r] = \pi(x+1, \infty) \tag{9.1}$$

We apply the similar condition at all nodes of the decision tree. Note that there exists a μ satisfying all these conditions: the space of queries is partitioned recursively between the left and right subtrees, so defining the probability of the left and right subspace at all nodes is a definition of μ (but note that μ with these properties need not be unique).

From (9.1), we can compute the information revealed about the query:

$$\begin{aligned} I_Q(\ell) - I_Q(\lambda) &= \log_2 \tfrac{\Pr[Q_\lambda]}{\Pr[Q_\ell]} = \log_2 \tfrac{1}{\pi(-\infty : x+1)} \\ I_Q(r) - I_Q(\lambda) &= \log_2 \tfrac{1}{\pi(x+1 : \infty)} \end{aligned}$$

Remember that our rule for a good separator was $\Delta I_Q \leq \rho \cdot \Delta H_N$. On the left side,

$I_Q(\ell) - I_Q(\lambda) = H_N(\lambda) - H_N(\ell)$, so the rule is trivially satisfied. On the right, the rule asks that: $\log_2 \frac{1}{\pi(x+1:\infty)} \leq \rho \cdot \log_2 \frac{1}{\pi(x-1:\infty)}$. Thus, x is a good separator iff $\pi(x+1:\infty) \geq \big[\pi(x-1:\infty)\big]^\rho$.

Finale. As defined above, a good separator satisfies the bound on the information progress, and guarantees the desired space bound of $O(n^\rho)$. We now ask what happens when no good separator exists.

We may assume by translation that the median of π is 0, so $\pi([1:\infty]) \leq \frac{1}{2}$. If $x = 1\frac{1}{2}$ is not a good separator, it means that $\pi(3:\infty) < \big[\pi(1:\infty)\big]^\rho \leq 2^{-\rho}$. If $x = 3\frac{1}{2}$ is not a good separator, then $\pi(5:\infty) < \big[\pi(3:\infty)\big]^\rho \leq 2^{-\rho^2}$. By induction, the lack of a good separator implies that $\pi(2j+1:\infty) < 2^{-\rho^j}$. The reasoning works symmetrically to negative values, so $\pi(-\infty:-2j-1) < 2^{-\rho^j}$.

Thus, if no good separator exists on coordinate i, the distribution of the values on that coordinate is very concentrated around the median. In particular, only a fraction of $\frac{1}{2d}$ of the database points can have $|x_i| \geq R = 1 + 2\log_\rho \log_2 \frac{n}{4d}$. Since there is no good separator on any coordinate, it follows that less than $d \cdot \frac{n}{2d} = \frac{n}{2}$ points have *some* coordinate exceeding R. Let $D^\star$ be the set of such database points.

To handle the case when no good separator exists, we can introduce a different type of node in the decision tree. This node tests whether the query lies in an ℓ_∞ ball of radius $R+1$ (which is equivalent to d coordinate comparisons). If it does, the decision tree simply outputs any point in $D \setminus D^\star$. Such a point must be within distance $2R+1$ of the query, so it is an $O(\log_\rho \log d)$ approximation.

If the query is outside the ball of radius $R+1$, a near neighbor must be outside the ball of radius R, i.e. must be in $D^\star$. We continue with the recursive construction of a decision tree for point set $D^\star$. Since $|D^\star| \leq |D|/2$, we get a one-bit reduction in the entropy of the answer for free. (Formally, our μ just assigns probability one to the query being outside the ball of radius $R+1$, because in the "inside" case the query algorithm terminates immediately.)

Intuition for a lower bound. After obtaining this information-theoretic understanding of the algorithm from [Ind98], the path to a lower bound should be intuitive. We will consider a distribution on coordinates decaying like $2^{-\rho^j}$ (we are free to consider only the right half, making all coordinates positive). Database points will be generated i.i.d., with each coordinate drawn independently from this distribution.

In the communication view, Alice's message sends a certain amount of information restricting the query space to some Q. The entropy of the answer is given by the measure of $Q + [-1,1]^d$ (each of the n points lands in $Q + [-1,1]^d$ independently with the same probability). The question that must be answered is how much bigger $Q + [-1,1]^d$ is, compared to Q. We show an isoperimetric inequality proving that the least expanding sets are exactly the ones generated by the algorithm from [Ind98]: intersections of coordinate cuts $\{p_i \geq x\}$.

Then, if Alice's message has $o(\rho \lg n)$ bits of information, the entropy of the near neighbor decreases by $o(\lg n)$ bits. In other words, $n^{1-o(1)}$ of the points are still candidate near neighbors, and we can use this to lower bound the message that Bob must send.

9.2 The Communication Lower Bound

We denote the communication problem c-NN by the partial function F. We complete the function F by setting $\bar{F}(q,D) = F(q,D)$ whenever $F(q,D)$ is defined (i.e., when we are either in a yes or no instance), and $\bar{F}(q, D) = \star$ otherwise. Note that the domain of $\bar{F}$ is $X \times Y$, where $X = \{0, 1, \ldots m\}^d$ and $Y = \left(\{0, 1, \ldots m\}^d\right)^n$.

An $[a, b]$-protocol is a protocol by which Alice sends a total of a bits and Bob sends a total of b bits. To prove Theorem 9.0.12, assume that there exists some $[a, b]$-protocol Π computing the function $\bar{F} : X \times Y \to \{0, 1, \star\}$.

As explained already, our lower bound only applies to deterministic (zero error) protocols. However, at many stages it requires conceptual use of distributions on the input domains X and Y, which are described below. We then use the richness lemma of [MNSW98] (for randomized protocols), namely Lemma 8.1.7.

First define the following measure (probability distribution) π over the set $\{0, 1, \ldots m\}$: for $i = 1, 2, \ldots c$, let $\pi(\{i\}) = 2^{-(2\rho)^i}$ and $\pi(\{0\}) = 1 - \sum_{i \geq 1} \pi(\{i\}) \geq 1/2$. For simplicity, we denote $\pi_i = \pi(\{i\})$. Similarly, define the measure μ_d over $\{0, 1, \ldots m\}^d$ as $\mu_d(\{(x_1, x_2 \ldots x_d)\}) = \pi(\{x_1\}) \cdot \pi(\{x_2\}) \cdots \pi(\{x_d\})$.

In our hard distribution, we generate q at random from $\{0, 1, \ldots m\}^d$ according to the distribution μ_d. Also, we take the set D by choosing n points i.i.d. from μ_d.

Claim 9.2.1. *If we choose q and D as above, then* $\Pr[\bar{F}(q, D) \neq 0] \leq e^{-\log^{1+\epsilon/3} n}$.

Proof. Consider q and some $p \in D$: they differ in the j^{th} coordinate by at least c with probability at least $2\pi_0\pi_c \geq \pi_c$ (when one is 0 and the other is c). Thus, $\Pr[\|q - p\|_\infty < c] \leq (1 - \pi_c)^d \leq e^{-\pi_c d} \leq e^{-\log^{1+\epsilon/2} n}$. By a union bound over all $p \in D$, we get that $\|q - p\|_\infty \geq c$ for all $p \in D$ with probability at least $1 - e^{-\log^{1+\epsilon/3} n}$. □

Claim 9.2.2. *There exists a combinatorial rectangle $\mathcal{Q} \times \mathcal{D} \subset \{0, 1, \ldots m\}^d \times (\{0, 1, \ldots m\}^d)^n$ on which the presumed protocol outputs 0, and such that $\mu_d(\mathcal{Q}) \geq 2^{-O(a)}$ and $\mu_{d \cdot n}(\mathcal{D}) \geq 2^{-O(a+b)}$.*

The claim follows immediately from the richness lemma 8.1.7, applied to the function F' that is the function the presumed protocol Π actually computes. In particular, note that since the protocol is deterministic, $F'(q, D) = \bar{F}(q, D)$ whenever $\bar{F}(q, D) \in \{0, 1\}$, and $F'(q, D)$ is either 0 or 1 when $\bar{F}(q, D) = \star$.

Since the protocol computes all of $\mathcal{Q} \times \mathcal{D}$ correctly, it must be that $\bar{F}(q, D) \in \{0, \star\}$ for all $q \in \mathcal{Q}$ and $D \in \mathcal{D}$. It remains to prove the following claim.

Claim 9.2.3. *Consider any set $\mathcal{Q} \subseteq \{0, 1, \ldots m\}^d$ and $\mathcal{D} \subseteq (\{0, 1, \ldots m\}^d)^n$ of size $\mu_d(\mathcal{Q}) \geq 2^{-\delta\rho \log n}$ and $\mu_{d \cdot n}(\mathcal{D}) \geq 2^{-O(n^{1-\delta})}$. Then, there exists some $q \in \mathcal{Q}$ and $D \in \mathcal{D}$ such that $\bar{F}(q, D) = 1$ (i.e., there exists a point $p \in D$ such that $\|q - p\|_\infty \leq 1$).*

The claim is based on the following lemma that we prove in Section 9.3. This lemma is a somewhat involved isoperimetric inequality on space with our distributions, and it is the core component of our lower bound.

Lemma 9.2.4. *Consider any set $S \subseteq \{0, 1, \ldots m\}^d$. Let $N(S)$ be the set of points at distance at most 1 from S under ℓ_∞: $N(S) = \{p \mid \exists s \in S : \|p - s\|_\infty \leq 1\}$. Then $\mu_d(N(S)) \geq (\mu_d(S))^{1/\rho}$.*

Proof of Claim 9.2.3. Let $N = N(\mathcal{Q})$ be the set of points at distance at most 1 from $\mathcal{Q}$. By the above lemma, $\mu_d(N) \geq (\mu_d(\mathcal{Q}))^{1/\rho} \geq 1/n^\delta$. We need to prove that there exists a set $D \in \mathcal{D}$ that intersects with N. For $D \in (\{0, 1, \ldots m\}^d)^n$, let $\sigma(D) = |D \cap N|$.

Suppose D would be chosen at random from $(\{0,1,\dots m\}^d)^n$ (instead of $\mathcal{D}$). Then $\mathbb{E}_D[\sigma(D)] \geq n \cdot n^{-\delta} = n^{1-\delta}$. By Chernoff bound, $\sigma(D) < 1$ happens only with probability at most $e^{-\Omega(n^{1-\delta})}$.

Thus, if we restrict to $D \in \mathcal{D}$, we obtain $\Pr_D[\sigma(D) < 1 \mid D \in \mathcal{D}] \leq \frac{\Pr_D[\sigma(D)<1]}{\Pr[D\in\mathcal{D}]} = e^{-\Omega(n^{1-\delta})} \cdot 2^{O(n^{1-\delta})} < e^{-\Omega(n^{1-\delta})}$.

Concluding, there exists some $D \in \mathcal{D}$ such that $|N(\mathcal{Q}) \cap D| \geq 1$, and thus there exists some $q \in \mathcal{Q}$ and some $p \in D$ such that $\|q - p\|_\infty \leq 1$. $\square$

Finally, Claims 9.2.2 and 9.2.3 imply that either $a = \Omega(\delta\rho \log n)$ or $b = \Omega(n^{1-\delta})$. This concludes the proof of Theorem 9.0.12.

9.3 An Isoperimetric Inequality: Proof of Lemma 9.2.4

We now prove Lemma 9.2.4.

The core of the lemma is the following one-dimensional isoperimetic inequality. The rest of the Lemma 9.2.4 results by an induction on the dimension.

Theorem 9.3.1. *Let ρ be a large positive integer, and for $i = 1 \dots m$, $\pi_i = 2^{-(2\rho)^i}$, $\pi_0 = 1 - (\pi_1 + \cdots + \pi_m)$. Then for any non-negative real numbers $\beta_0, \dots, \beta_m$ satisfying*

$$\pi_0\beta_0^\rho + \pi_1\beta_1^\rho + \cdots + \pi_m\beta_m^\rho = 1$$

the following inequality holds (where we set $\beta_{-1} = \beta_{m+1} = 0$)

$$\sum_{i=0}^{m} \pi_i \max\{\beta_{i-1},\ \beta_i,\ \beta_{i+1}\} \quad \geq \quad 1. \tag{9.2}$$

Before proving Theorem 9.3.1, we complete the proof of Lemma 9.2.4 assuming this one-dimensional theorem. Let's prove first the case of $d = 1$. We have a set $S \subset \{0,1,\dots m\}$, and let $\beta_i = 1$ iff $i \in S$. Then $\mu_1(S) = \pi(S) = \sum \pi_i\beta_i = \sum \pi_i\beta_i^\rho$. The set $N(S) = \Big\{i \in \{0,1,\dots m\} \mid \max\{\beta_{i-1},\beta_i,\beta_{i+1}\} = 1\Big\}$ has measure, by Theorem 9.3.1,

$$\mu_1(N(S)) = \sum_{i=0}^{m} \pi_i \max\{\beta_{i-1},\beta_i,\beta_{i+1}\} \geq (\mu_1(S))^{1/\rho}.$$

Now let's prove the induction step. Consider $S \subset \{0,1,\dots m\}^d$, and, for $i \in \{0,1,\dots m\}$, let $S_i = \{(s_2,s_3,\dots s_d) \mid (i,s_2,\dots s_m) \in S\}$ be the set of points in S that have the first coordinate equal to i. Then, letting $\beta_i^\rho = \mu_{d-1}(S_i)$, we have that

$$\sum_{i=0}^{m} \pi_i\beta_i^p = \sum_i \pi_i\mu_{d-1}(S_i) = \mu_d(S).$$

We can lower bound the measure of $N(S)$ as

$$\mu_d(N(S)) \geq \sum_{i=0}^{m} \pi_i \cdot \max\begin{cases} \mu_{d-1}(N(S_{i-1})) \\ \mu_{d-1}(N(S_i)) \\ \mu_{d-1}(N(S_{i-1})) \end{cases}$$

where we assume, by convention, that $S_{-1} = S_{m+1} = 0$.

By inductive hypothesis, $\mu_{d-1}(N(S_i)) \geq (\mu_{d-1}(S_i))^{1/\rho} = \beta_i$ for all i. Thus, applying Theorem 9.3.1 once again, we conclude

$$\mu_d(N(S)) \geq \sum_i \pi_i \max\{\beta_{i-1}, \beta_i, \beta_{i+1}\} \geq (\mu_d(S))^{1/\rho}.$$

This finishes the proof of Lemma 9.2.4.

9.3.1 The 1D case: Proof of Theorem 9.3.1

Let $\Gamma = \left\{(\beta_0, \dots, \beta_m) \in \mathbb{R}^{m+1} \mid \pi_0\beta_0^\rho + \pi_1\beta_1^\rho + \cdots + \pi_m\beta_m^\rho = 1\right\}$, and denote by $f(\beta_0, \dots, \beta_m)$ the left hand side of (9.2). Then f is a continuous function on the compact set $\Gamma \subset \mathbb{R}^{m+1}$, so it achieves its minimum. Call an $(m+1)$-tuple $(\beta_0, \dots, \beta_m) \in \Gamma$ *optimal* if $f(\beta_0, \dots, \beta_m)$ is minimal. Our proof strategy will be to show that if $(\beta_0, \dots, \beta_m)$ is optimal, then $\beta_i = 1$.

We consider several possible configurations for sizes of β_i's in an optimal β in three separate lemmas, and prove they are not possible. We then conclude the theorem by showing these configurations are all the configurations that we need to consider.

Lemma 9.3.2. *If there exists an index $i \in \{1, \dots, m-1\}$ such that $\beta_{i-1} > \beta_i < \beta_{i+1}$, then $\bar{\beta} = (\beta_0, \dots, \beta_m)$ is not optimal.*

Proof. Define a new vector $\bar{\beta}' = (\beta_0, \dots, \beta_{i-2}, \beta_{i-1} - \epsilon, \beta_i + \delta, \beta_{i+1} - \epsilon, \beta_{i+2}, \dots, \beta_m)$, where $\epsilon,\ \delta > 0$ are chosen suitably so that $\bar{\beta}' \in \Gamma$, and $\beta_{i-1} - \epsilon > \beta_i + \delta < \beta_{i+1} - \epsilon$. It's easy to see that $f\left(\bar{\beta}\right) > f\left(\bar{\beta}'\right)$, which contradicts the optimality of $\bar{\beta}$. $\square$

Lemma 9.3.3. *If there exists an index $i \in \{1, \dots, m\}$ such that $\beta_{i-1} > \beta_i \geq \beta_{i+1}$, then $\bar{\beta} =(\beta_0, \dots, \beta_m)$ is not optimal.*

Proof. Let $\beta = \left(\frac{\pi_{i-1}\beta_{i-1}^\rho + \pi_i\beta_i^\rho}{\pi_{i-1}+\pi_i}\right)^{1/\rho}$ and define $\bar{\beta}' = (\beta_0, \dots, \beta_{i-2}, \beta, \beta, \beta_{i+1}, \dots \beta_m)$. Then $\bar{\beta}' \in \Gamma$, and $\beta_{i-1} > \beta > \beta_i$.

We claim that $f(\bar{\beta}) > f(\bar{\beta}')$. Comparing the expressions for $f\left(\bar{\beta}\right)$ and $f\left(\bar{\beta}'\right)$ term by term, we see that it's enough to check that

$$\begin{aligned}\pi_i \max\{\beta_{i-1}, \beta_i, \beta_{i+1}\} + \pi_{i+1} \max\{\beta_i, \beta_{i+1}, \beta_{i+2}\} \quad &> \\ > \pi_i \max\{\beta, \beta_{i+1}\} + \pi_{i+1} \max\{\beta, \beta_{i+1}, \beta_{i+2}\}&\end{aligned}$$

where the terms involving π_{i+1} appear unless $i = m$. For $i = m$, the inequality becomes $\beta_{i-1} > \beta$ which holds by assumption. For $i = 1, \dots, m-1$, the above inequality is equivalent to

$$\pi_i(\beta_{i-1} - \beta) > \pi_{i+1} \cdot (\max\{\beta, \beta_{i+2}\} - \max\{\beta_i, \beta_{i+2}\})$$

which, in its strongest form (when $\beta_i \geq \beta_{i+2}$), is equivalent to $\pi_i(\beta_{i-1} - \beta) > \pi_{i+1}(\beta - \beta_i)$. The last inequality is equivalent to

$$\left(\frac{\pi_i\beta_{i-1} + \pi_{i+1}\beta_i}{\pi_i + \pi_{i+1}}\right)^\rho > \frac{\pi_{i-1}\beta_{i-1}^\rho + \pi_i\beta_i^\rho}{\pi_{i-1} + \pi_i}$$

which we can rewrite as

$$\left(\frac{c_i + t}{c_i + 1}\right)^\rho - \frac{c_{i-1} + t^\rho}{c_{i-1} + 1} > 0, \tag{9.3}$$

where $c_i = \pi_i/\pi_{i+1} \geq 2^{(2\rho)^{i+1}-(2\rho)^i}$ (for $i > 0$ we have equality and for $i = 0$ we have inequality because p is large), and $t = \beta_i/\beta_{i-1} \in [0, 1)$. Let $F(t)$ denote the left hand side of inequality (9.3) (which we are left to prove). Note that $F(0) > 0$, because:

$$\left(\frac{c_i}{c_i+1}\right)^\rho = \left(1 - \frac{1}{c_i+1}\right)^\rho \;\geq\; 1 - \frac{\rho}{c_i+1} > 1 - \frac{1}{c_{i-1}+1} \;=\; \frac{c_{i-1}}{c_{i-1}+1}$$

where we have used Bernoulli's inequality $(1-x)^n \geq 1 - nx$ for $0 < x < 1/n$ and $c_i + 1 > 2^{(2\rho)^{i+1}-(2\rho)^i} > \rho \cdot (2^{(2\rho)^i} + 1) = \rho(\frac{1}{\pi_{i-1}} c_{i-1} + 1) > \rho(c_{i-1} + 1)$. Now we let $t \in (0, 1)$ and write $F(t) = F(0) + t^\rho G(t)$, where

$$\begin{aligned} G(t) \quad &= \quad \frac{1}{(c_i+1)^\rho}\left(\binom{\rho}{1} c_i^{\rho-1}\frac{1}{t} + \binom{\rho}{2} c_i^{\rho-2}\frac{1}{t^2} + \cdots + \binom{\rho}{\rho-1} c_i \frac{1}{t^{\rho-1}}\right) + \\ &\quad + \left(\frac{1}{(c_i+1)^\rho} - \frac{1}{c_{i-1}+1}\right). \end{aligned}$$

If $G(t) \geq 0$, then clearly $F(t) \geq F(0) > 0$, so we are done. Otherwise, $G(t) < 0$, and in this case it easily follows that $G(1) < G(t) < 0$, hence $F(t) = F(0) + t^\rho G(t) > F(0) + G(1) = F(1) = 0$, as desired. This concludes the proof of the lemma. $\square$

Lemma 9.3.4. *If there is an index $i \in \{0, 1 \ldots, m-1\}$ such that $\beta_{i-1} \leq \beta_i < \beta_{i+1}$, then $\beta = (\beta_0, \beta_1, \ldots, \beta_m)$ is not optimal.*

Proof. We proceed as in the previous lemma. Let $\beta = \left(\frac{\pi_i\beta_i^\rho + \pi_{i+1}\beta_{i+1}^\rho}{\pi_i + \pi_{i+1}}\right)^{1/\rho}$, and define $\bar{\beta}' = (\beta_0, \ldots, \beta_{i-1}, \beta, \beta, \beta_{i+2}, \ldots, \beta_m)$. As before, $\bar{\beta}' \in \Gamma$ and $\beta_i < \beta < \beta_{i+1}$. We claim that $f(\bar{\beta}) > f(\bar{\beta}')$. Comparing the expressions for $f\left(\bar{\beta}\right)$ and $f\left(\bar{\beta}'\right)$ term by term, we see that it's enough to check that

$$\begin{aligned} &\pi_{i-1} \max\{\beta_{i-2}, \beta_{i-1}, \beta_i\} + \pi_i \max\{\beta_{i-1}, \beta_i, \beta_{i+1}\} \quad > \\ &\quad > \pi_{i-1} \max\{\beta_{i-2}, \beta_{i-1}, \beta\} + \pi_i \max\{\beta_{i-1}, \beta, \beta\} \end{aligned}$$

where the terms involving π_{i-1} appear unless $i = 0$. If $i = 0$, the above inequality becomes $\beta_{i+1} > \beta$ and we are done. For $i = 1, \ldots m-1$, the inequality is equivalent to

$$\pi_i(\beta_{i+1} - \beta) > \pi_{i-1} \cdot (\max\{\beta, \beta_{i-2}\} - \max\{\beta_i, \beta_{i-2}\})$$

which, in its strongest form (when $\beta_i \geq \beta_{i-2}$) is equivalent to $\pi_i(\beta_{i+1} - \beta) > \pi_{i-1}(\beta - \beta_i)$. The latter inequality is equivalent to

$$\left(\frac{\pi_i\beta_{i+1} + \pi_{i-1}\beta_i}{\pi_i + \pi_{i-1}}\right)^\rho > \frac{\pi_{i+1}\beta_{i+1}^\rho + \pi_i\beta_i^\rho}{\pi_{i+1} + \pi_i}$$

which we can rewrite as

$$\left(\frac{c_{i-1}t+1}{c_{i-1}+1}\right)^\rho - \frac{c_i t^\rho + 1}{c_i + 1} > 0, \tag{9.4}$$

where $c_i = \pi_i/\pi_{i+1}$ as before, and $t = \beta_i/\beta_{i+1} \in [0, 1)$. Let $F(t)$ denote the left hand side of

(9.4) (which we are left to prove). Note that $F(0) > 0$, because

$$\begin{aligned}\left(\frac{1}{c_{i-1}+1}\right)^{\rho} &> \frac{1}{(2c_{i-1})^{\rho}} = \frac{1}{\pi_{i-1}^{\rho}} \cdot 2^{-\rho\cdot(2\rho)^i-\rho} \\ &> 2^{-\rho\cdot(2\rho)^i-\rho} \geq 2^{(2\rho)^i-(2\rho)^{i+1}} = \frac{1}{c_i} > \frac{1}{c_i+1}\end{aligned}$$

Now we let $t \in (0,1)$ and write $F(t) = F(0) + t^{\rho}G(t)$, where

$$\begin{aligned} G(t) &= \frac{1}{(c_{i-1}+1)^{\rho}}\left(\binom{\rho}{1}c_{i-1}\frac{1}{t} + \binom{\rho}{2}c_{i-1}^2\frac{1}{t^2} + \cdots + \binom{\rho}{\rho-1}c_{i-1}^{\rho-1}\frac{1}{t^{\rho-1}}\right) + \\ &\quad + \left(\left(\frac{c_{i-1}}{c_{i-1}+1}\right)^{\rho} - \frac{c_i}{c_{i-1}+1}\right).\end{aligned}$$

If $G(t) \geq 0$, then clearly $F(t) \geq F(0) > 0$, so we are done. Otherwise, $G(t) < 0$, in which case it easily follows that $G(1) < G(t) < 0$, hence $F(t) = F(0)+t^{\rho}G(t) > F(0)+G(1) = F(1) = 0$, as desired. This concludes the proof of the lemma. □

To prove Theorem 9.3.1, assume $\bar{\beta} = (\beta_0, \ldots, \beta_m) \in \Gamma$ is optimal. By Lemmas 9.3.2 and 9.3.3, it follows that $\beta_0 \leq \beta_1 \leq \cdots \leq \beta_m$. Now Lemma 9.3.4 implies that $\beta_0 = \beta_1 = \cdots = \beta_m$, so since $\bar{\beta} \in \Gamma$, we have $\beta_i = 1$, and hence the minimal value of f over $\Gamma$is $f(1,1,\ldots,1) = 1$.

This concludes the proof of the Theorem 9.3.1.

9.4 Lower Bounds for NN with a High Approximation

In this section, we present an argument why it is difficult to prove any non-trivial lower bounds for randomized NN problems for high approximation. Namely, we show that the current techniques are not able to prove communication complexity lower bounds for randomized NN problems for an approximation bigger than 3. The approximation factor of 3 seems to be fundamental here. For approximation less than 3, we actually know lower bounds for NN under ℓ_∞, by reduction to the partial match problem.

Our arguments apply to NN over any metric. Let us consider a metric $\mathcal{M}$ with distance function $\mathsf{d}_{\mathcal{M}}$ and the following problem.

Definition 9.4.1 (NN under $\mathcal{M}$)**.** *Fix $R > 0, \alpha > 0$. Suppose Alice is given a point $q \in \mathcal{M}$, and Bob is given the dataset $D \subset \mathcal{M}$ of size n. Then, in the R-*Near Neighbor Search *problem, Alice and Bob compute the following function $\mathfrak{N}(q, D)$:*

- $\mathfrak{N}(q, D) = 1$ *if there exists some $p \in D$ such that $\mathsf{d}_{\mathcal{M}}(x, y) \leq R$;*
- $\mathfrak{N}(q, D) = 0$ *if for all $p \in D$, we have that $\mathsf{d}_{\mathcal{M}}(x, t) \geq \alpha R$.*

As before, when neither is the case, we set $\mathfrak{N}(x, y) = \star$.

In a randomized $[a, b]$-protocol Π, Alice sends at most a bits, Bob sends at most b bits, and they produce the correct answer with probability at least 0.9.

An almost ubiquitous technique to prove a lower bound for the communication complexity is by applying Yao's minimax principle. The principle says that if there exists a randomized $[a, b]$-protocol, then for any distribution μ on $\mathcal{M} \times \mathcal{M}^n$, there exists some deterministic protocol Π_μ succeeding on 0.9 mass of the distribution μ: $\mathbb{E}_{(q,D)\in\mu}[\Pi_\mu(x,y) = \mathfrak{N}(x,y)] \geq 0.9$ Thus one just need to exhibit a "hard" distribution where no deterministic protocol succeeds. Most candidates for the "hard" distribution μ are *product distributions*, namely $\mu = \mu_q \times \mu_D$, where μ_q and μ_D are independent distributions on $q \in \mathcal{M}$ and $D \in \mathcal{M}^n$ respectively.

Indeed, to the best of our knowledge, all known asymmetric communication complexity lower bounds are proven via this approach with product distributions. It seems quite challenging to prove asymmetric communication complexity lower bounds for distributions that are non-product.

We prove that it is not possible to prove lower bound for the NN problem with product distributions when the approximation is bigger than 3. In fact, the argument applies even to one-way protocol lower bounds, where one-way protocols are $[a, 0]$-protocols in which just Alice sends a message of length a.

Lemma 9.4.2. *Consider the problem $\mathfrak{N}$ for approximation α. Consider any product distribution $\mu = \mu_q \times \mu_D$ on $\mathcal{M} \times \mathcal{M}^n$, and suppose for any $[a, 0]$-protocol Π, we have $\mathbb{E}_\mu[\Pi(x,y) \neq \mathfrak{N}(x,y)] < 0.9$. Then either $\alpha \leq 3$ or $a = O(\log n)$ or there exists (q, D) in the support of μ such that $\mathfrak{N}(q, D) = \star$.*

Proof. Assume that $\alpha > 3$ and that $a \geq C \log n$ for some big constant C. Let $\mathcal{Q}$ be the support of μ_q and $\mathcal{D}$ be the support of μ_D. We will prove that there exists some $(\tilde{q}, \tilde{D}) \in \mathcal{Q} \times \mathcal{D}$ such that $\mathfrak{N}(\tilde{q}, \tilde{D}) = \star$.

We will use a characterization of [KNR99] for one-way protocols for product distributions to construct $\tilde{q}, \tilde{D}$.

First we need to give a definition. Consider the matrix M of size $|\mathcal{Q}| \times |\mathcal{D}|$ where $M_{ij} = \mathfrak{N}(q_i, D_j)$, where q_i is the i^{th} element of $\mathcal{Q}$ in, say, lexicographic order, and same with D_j. The *VC-dimension* of M is the maximum $v \in \mathbb{N}$ such that there exists $D_{j_1}, \dots D_{j_v} \in \mathcal{D}$ such that for any boolean vector $z \in \{0,1\}^v$, there exist $q_z \in \mathcal{Q}$ with $\mathfrak{N}(q_z, D_{j_k}) = z_k$ for all $k \in [v]$.

Since $a \geq C \log n$, the result of [KNR99] implies that the VC-dimension of the matrix M is at least $v \geq \log_2 n + 2$ (choosing C accordingly). Then, take a set of z's that is $Z \subset \{1\} \times \{0,1\}^{v-1}$ and has size $|Z| \geq n+1$. Suppose $Z = \{z^{(1)}, z^{(2)}, \dots, z^{(n+1)}\}$ and let $q_{z_1} \dots q_{z_{n+1}}$ be the queries such that, for all $i = 1 \dots n+1$, we have that $\mathfrak{N}(q_{z^{(i)}}, D_{j_k}) = z_k^{(i)}$ for all $k \in [v]$. In particular, for $D = D_{j_1}$, we have that $\mathfrak{N}(q_z, D) = 1$, i.e., there exists $p_z \in D$, for each $z \in Z$, such that $\mathsf{d}_\mathcal{M}(q_z, p_z) \leq R$. By pigeonhole principle, there exists some $p \in D$ and distinct $z', z'' \in Z$ such that $\mathsf{d}_\mathcal{M}(q_{z'}, p) \leq R$ and $\mathsf{d}_\mathcal{M}(q_{z''}, p) \leq R$. Thus, by triangle inequality, $\mathsf{d}_\mathcal{M}(q_{z'}, q_{z''}) \leq 2R$. However, since $z' \neq z''$, there is some $j \in \{2, \dots v\}$ such that $z'_j \neq z''_j$. In other words, wlog, $\mathsf{d}_\mathcal{M}(q_{z'}, D_j) \leq R$ and $\mathsf{d}_\mathcal{M}(q_{z''}, D_j) \geq \alpha R > 3R$. But this is not possible since, by triangle inequality, $\mathsf{d}_\mathcal{M}(q_{z''}, D_j) \leq \mathsf{d}_\mathcal{M}(q_{z''}, q_{z'}) + \mathsf{d}_\mathcal{M}(q_{z'}, D_j) \leq 2R + R = 3R$ — a contradiction. ☐

9.5 Decision Trees Lower Bound

We formally define what we mean by a decision tree for a data structure problem (see also [KN97]). Consider a partial problem $F : \mathcal{I} \to \{0,1\}$ with $\mathcal{I} \subset X \times Y$, where X is the set of "queries" and Y is the set of "datasets".

For $y \in Y$, a *decision tree* T_y is a complete binary tree in which:

- each internal node v is labeled with a predicate function $f_v : X \to \{0,1\}$. We assume f_v comes from some set $\mathcal{F}$ of *allowed predicates*.

- each edge is labeled with 0 or 1, indicating the answer to the parent's predicate.
- each leaf is labeled with 0 or 1, indicating the outcome of the computation.

Evaluating T_y on x is done by computing the root's predicate on x, following the corresponding edge, computing the next node's predicate, and so on until a leaf is reached. The label of the leaf is the output, denoted $T_y(x)$.

We let the *size* s of the tree to be the total number of the nodes. The *depth* d of the tree is the longest path from the root to a leaf. The *predicate size* is $w = \lceil \log_2 \mathcal{F} \rceil$.

We say that problem F can be solved by a decision tree of size s, depth d, and predicate size w iff, for any y, there exists some tree T_y of size at most s, depth at most d, and node size at most w, such that $T_y(x) = F(x, y)$ whenever $(x, y) \in \mathcal{I}$.

Our result on the decision tree lower bound follows from the following folklore lemma, which converts an efficient decision tree solving a problem F into an efficient communication protocol.

Lemma 9.5.1. *Consider any (promise) problem $F : \mathcal{I} \to \{0, 1\}$, where $\mathcal{I} \subset X \times Y$. Suppose there exists a decision tree of size s, depth d, and node size w.*

If Alice receives $x \in X$ and Bob receives $y \in Y$, there exists a communication protocol solving the problem F, in which Alice sends a total of $a = O(\log s)$ bits and Bob sends $b = O(dw \log s)$ bits.

Proof. Before the protocol, Bob constructs his decision tree T_y. Suppose, for a moment, that the decision tree is balanced, that is $d = O(\log s)$. Then, Alice and Bob can run the following "ideal" protocol. In round one, Bob sends the predicate f_r of the root r of the decision tree. Alice computes $f_r(x)$ (a bit) and sends it back. Then Bob follows the corresponding edge in the tree, and sends the predicate of the corresponding child, etc. We obtain communication $a \le d$ and $b \le w \cdot d$.

In general, however, the decision tree T_D is not balanced. In this case, Alice and Bob can simulate a standard binary search on a tree. Specifically, Bob finds a separator edge that splits the tree in two components, each of size at least $s/3$. Let this separating edge be (u, v). In round one, Alice and Bob want to detect whether, in the ideal protocol, Alice would eventually follow the edge (u, v). To determine this, Bob sends the predicates for all nodes on the path from the root r to u. Alice evaluates these predicates on x and sends back a 1 if she would follow the edge (u, v), and 0 otherwise. Then, the players recurse on the remaining part of the tree; they are done after $O(\log s)$ such rounds.

In the end, Alice sends only $a = O(\log s)$ bits, i.e. one bit per round. Bob sends $O(d \cdot w)$ bits per round, and thus $b = O(dw \log s)$. □

Chapter 10

Conclusions

In this thesis, we presented new efficient algorithms for the nearest neighbor problem and related tasks, obtained via old approaches as well as new ones. We also showed impossibility results, pointing out the limits of the NN algorithms. Our contributions paint a more complete and multifaceted landscape of the NN problem(s).

Our specific contributions are as follows. First, we designed a new algorithm for the NN problem under the standard Euclidean distance. Our algorithm is based on the general technique of Locality-Sensitive Hashing. Besides significantly improving over the state-of-art algorithm, our new algorithm is, in fact, *near-optimal* in the class of LSH-based algorithms.

In the next step we focused our attention to NN under other metrics, most notably the string edit distance. We showed that, for this problem, the classical approaches — such as embedding into ℓ_1, powers of ℓ_2, or using small–space sketches — *fundamentally cannot achieve* an approximation factor below a near-logarithmic barrier. Our lower bound also applies to an important variant of the edit distance, the Ulam distance, as well the Earth-Mover Distance (EMD) over the hypercube.

Motivated by the above conclusion, we proposed a new approach to designing NN algorithms, via embeddings into (iterated) product spaces. As a "proof of concept", we showed a new NN algorithm for the Ulam distance that overcomes the aforementioned barrier. Our NN algorithm achieves a double-logarithmic approximation, which is an *exponential improvement* over of the classical approaches.

At the same time, we demonstrated that our algorithmic tools, from both the old approaches and the new ones, extend to other problems as well. Specifically, we showed that the LSH technique itself is useful for approximating kernel spaces. Furthermore, the concept of embedding into product spaces was instrumental in designing algorithms on computing edit distances. In one instance, we designed a near-linear time algorithm for the classical problem on computing the edit distance between two strings, with a greatly improved approximation factor.

Finally, we show several data structure lower bounds for the NN problem under standard Euclidean, Hamming, and ℓ_∞ distances. Our lower bounds are the *first* to hold in the same computational models as the respective algorithms, and, in fact, *match* the parameters of those algorithms.

10.1 Open Questions

Although many of our results are near-optimal, our results open many avenues for further exploration. Below we describe several intriguing questions that (we believe) hold they key to improving our understanding of the NN problem and related questions, from both practical and theoretical standpoints.

LSH-based NN algorithms. In Section 3.2.1, we showed the existence of hash functions which yield near-optimal exponent in the running time of the LSH algorithm. This greatly improves over the state-of-the-art hash functions from [DIIM04]. However, an important and fascinating open question remains to find a way to evaluate these new hash functions efficiently. If one were able to evaluate them in time comparable to that of functions from [DIIM04], this would lead to search procedures that are orders of magnitude faster than the current implementations of LSH (such as the E^2LSH package [AI05], based on [DIIM04]).

As we mentioned in Section 3.3.2, one way to obtain such faster hash functions is to use particular high-dimensional lattices. For example, we have investigated the Leech Lattice, a 24-dimensional mathematical structure well-studied in the information theory community. Finding its analogue for higher (or arbitrary) dimensions could provide the desired hash functions.

Product spaces for NN and other problems. Another exciting question is that of understanding the power of product spaces. At the moment, they remain quite poorly understood, both in their richness (what are they good for?), and tractability (can we design efficient algorithms for them?).

Motivated by the developments for the Ulam distance, it is only natural to ask: is it possible to embed the standard edit distance into some iterated product metric with a constant distortion? Such a result, combined with our NN algorithm from Chapter 5, could have far-reaching algorithmic implications, both in theory and in practice (see also Section 2.1).

We can also ask a similar embedding question for the planar EMD metric. Here, too, we know that classical approaches to the NN problem (embedding into standard spaces) meet a concrete non-approximability barrier, and bypassing it would be of big significance (see Section 2.1).

Besides the application of product spaces to NN, it is also interesting to understand the use of product spaces for other problems. For example, one particular question is to understand the sketching complexity of the product spaces. Very recently, partial progress has been made in [JW09, ADIW09].

From a more general viewpoint, we still lack technology to deal with the iterated product spaces, as compared to the well established theory of ℓ_2 or ℓ_1 spaces. For instance, we need tools to be able to prove non-embeddability results, or answer questions of the type "what is the 'simplest' iterated product space required for the Ulam distance?"

Furthermore, the appeal of the iterated product spaces comes also from the fact that they may be seen as computational primitives. In particular, one may view an iterated product space as a certain "circuit" of low-depth and polynomial size. For instance, the product space resulting from the Ulam distance embedding is a circuit of depth 3, where the lowest level has "ℓ_1" gates, the middle level has "ℓ_∞" gates, and the top-most level is just one "$(\ell_p)^p$" gate. Is there more to be gained from this analogy?

Edit and Ulam distances. Although not the immediate goal of this thesis, our results also provide a better understanding of the classical edit distance. Yet this still remains only the tip of the iceberg. There are many now-standard questions on edit distance, such as finding the best distortion embedding into ℓ_1 (see Section 1.3.2) and computing the distance in near-linear time (see Chapter 6). Above we also mentioned the question of embedding into iterated product spaces.

We bring forward another aspect: what is the exact relationship between the edit distance and the Ulam distance, with the latter being just the restriction of the former to non-repetitive strings? For one, for the Ulam distance, we know better algorithms than for the general edit distance — namely, for distance computation in near-linear time (via the "patience sorting" algorithm) and for NN with a double-logarithmic approximation (see Chapter 5). We also know that the edit distance (on binary strings) is no easier than the Ulam distance, at least up to constant approximation (see Chapter A).

Thus the following question emerges: is the edit distance (on binary strings) really harder than the Ulam distance? At the moment, we have no lower bound separating the two metrics. All the lower bounds for the edit distance on binary strings hold to (almost) the same degree as for the Ulam distance, including non-embeddability into ℓ_1 (see [KR06] and Chapter 7), or sketching lower bounds (see Chapter 7 and [AJP10]), or query complexity for sub-linear time algorithms (see [BEK$^+$03, AN10]). Thus, the only "provable hardness" of the edit distance is already intrinsic to the Ulam distance. Separating these two distances may pinpoint why the edit distance still lacks more efficient algorithms.

Sketching. We have shown sketching complexity lower bounds for the Ulam, edit, and EMD distances. As argued before, these provide more computational insight and robustness to the lower bounds as compared to the previous ℓ_1 non-embeddability results, which were of rather "geometric" flavor.

Nonetheless, it is natural to ask whether the two types of lower bounds — sketching lower bounds and non-embeddability into normed spaces — are really that different. In one direction, embedding into normed spaces implies efficient sketches (see Section 1.3.2). The question is whether a form of the inverse relation holds: that non-embeddability into normed spaces of a metric implies sketching lower bounds for that metric. Although such a hope might seem far-fetched, our characterization from Section 7.2 suggests this is not out of question, at least for restricted classes of metrics. Such a result would also be an important step towards a bigger goal of *characterizing sketchable metrics* (see, e.g., [McG07, GIM08]).

NN data structure lower bounds. An important research direction is establishing lower bounds for the NN algorithms under various metrics, such as the Euclidean (ℓ_2), the Hamming (ℓ_1), ℓ_∞, or the edit distances.

For example, for the Hamming and Euclidean distances, the main question is: are the LSH-based algorithms the best possible algorithms? In the regime of constant query time, our results from Chapter 8 and the recent result of [PTW08] show that this is indeed the case. A major question is to prove a $n^{\Omega(1/c^2)}$ query complexity lower bound, even in the regime of near-linear space. We note, however, that we currently lack techniques able to prove such lower bounds: we do not have super-logarithmic query lower bounds for *any* static data structure problem.

For the ℓ_∞ distance, a natural next step would be to prove a *randomized* lower bound for the communication complexity problem considered in Chapter 9. This already seems to highlight an interesting technical barrier we need to overcome. Specifically, most asymmetric communication complexity lower bounds consider a product distribution (including our lower bounds from Chapters 8 and 9). However, as we show in Section 9.4, such approach cannot generally be used for proving NN lower bounds for approximation $c > 3$. (We note that the result of [PTW08] does manage to surpass this barrier, albeit their lower bound holds for a slighter harder problem than the decision NN problem we consider in our NN lower bounds.)

We would also like to point out one curiosity that invites further investigation. Specifically, all (efficient) NN algorithms for the Euclidean and Hamming distances are randomized algorithms, and communication complexity lower bounds suggest that deterministic efficient algorithms might not be achievable (see [Liu04, PT06]). In contrast, for ℓ_∞, the only algorithm we know of is a deterministic algorithm, but which achieves worse approximation (double-logarithmic in the dimension, in the polynomial space regime). Furthermore, the two algorithms seem fundamentally different. Reconciling these differences is an interesting question to explore.

Appendix A

Reducing Ulam to Edit Distance on 0-1 strings

In this section we prove Lemma 2.1.1, namely that the Ulam distance is no harder than the edit distance on binary strings, up to a constant factor approximation. The key idea is that substituting every alphabet symbol independently with a random bit is likely to preserve the edit distance, up to constant factor.

The basic intuition behind this proof is quite simple. The first part (the upper bound on $\mathrm{ed}(\pi(P), \pi(Q))$) is immediate, and the main challenge is to prove the lower bound on $\mathrm{ed}(\pi(P), \pi(Q))$. To prove the lower bound, we proceed by ruling out all "potential certificates" that $\mathrm{ed}(\pi(P), \pi(Q))$ is small. Specifically, a "potential certificate" is a potential fixed alignment between $\pi(P)$ and $\pi(Q)$ of low cost, i.e. a fixed monotone mapping that matches monotonically all but at most $\frac{1}{100}\mathrm{ed}(P, Q)$ of the positions in $\pi(P)$ and $\pi(Q)$. We then analyze the probability that such an alignment is "successful", in the sense that every pair of positions that is matched under the potential alignment has equal symbols. Indeed, we show this probability is exponentially small because many of the pairs matched are independent coin tosses. We then apply a union bound over all potential alignment of small cost. Although a direct union bound is not sufficient (there are too many potential alignments to consider), we reduce the number of potential low-cost alignments by partitioning the set of all such alignments into a smaller number of groups of "equivalent alignments".

We proceed to set up some basic terminology and notation and to provide two lemmas that will be used in the proof of the lemma.

For two permutations P, Q, we say that an index (position) $i \in [d]$ in P is *missing* (from Q) if the symbol $P(i)$ does not appear inside Q.[1] We say that a pair of indices $\{i, j\} \subseteq [d]$ is an *inversion* (in P with respect to Q) if the two characters $P(i), P(j)$ appear in Q but in the opposite relative order than in P, formally given by $(i-j)(Q^{-1}(P(i)) - Q^{-1}(P(j))) < 0$. We also say that index j is *inverted* with respect to i.

An *alignment* of two strings $x, y \in \Sigma^d$ is a mapping $A : [d] \mapsto [d] \cup \{\perp\}$ that is monotonically increasing on $A^{-1}([d]) = \{i \in [d] \mid A(i) \in [d]\}$. Intuitively, A models a *candidate* longest common subsequence between x and y, and thus it maps indices in x to their respective indices in y and takes the value $\perp$ when there is no respective index in y (i.e., the respective position of x is not in the candidate subsequence). A *disagreement* in the alignment A is an index $i \in [d]$ for which $A(i) \neq \perp$ and $x(i) \neq y(A(i))$. The alignment is called

[1] Remember that we have defined a permutation P as a string with a large alphabet where every symbol appears at most once.

successful if it has no disagreements. The *cost* of an alignment is the number of positions in x (equivalently, in y) that are not mapped to a respective index in the other string, namely $|A^{-1}(\perp)| = d - |A^{-1}([d])| = d - |A([d])|$, where $A([d]) = \{A(i) \mid i \in [d]\}$. It is easy to verify that for all x, y,

$$\tfrac{1}{2}\,\mathrm{ed}(x,y) \le \min_A \mathrm{cost}(A) \le \mathrm{ed}(x,y), \tag{A.1}$$

where the minimum is taken over all successful alignments A.

In the following claim, we present a property of strings P and Q that will let us prove that, for a fixed potential alignment between $\pi(P)$ and $\pi(Q)$, the probability of the alignment being successful is very small.

Claim A.0.1. *Let P, Q be two permutations of length d that contain the same symbols, i.e. $P([d]) = Q([d])$. Then there exists a collection of $m \ge \mathrm{ed}(P,Q)/4$ inversions $\{i_1, j_1\}, \ldots, \{i_m, j_m\}$ such that $i_1, j_1, \ldots, i_m, j_m$ are all distinct.*

Proof. Fix P, Q. Define an (undirected) graph G with vertex set $[d]$ and an edge $\{i, j\}$ whenever $\{i, j\}$ is an inversion. Let $E^* \subseteq E(G)$ be a matching in G (i.e. no two edges in E^* share an endpoint) that is maximal with respect to containment. Observe that E^* is a collection of inversions whose indices are all distinct (as desired), and it only remains to bound $m = |E^*|$ from below. Following [SU04], we achieve the latter using the well-known relation between maximal matching and vertex-cover.[2]

Let V^* be the set of vertices incident to any edge in E^*, thus $|V^*| = 2|E^*|$. Clearly, V^* is a vertex-cover of G, namely every edge (inversion) must have at least one endpoint in V^*. It follows that $V \setminus V^*$ contains no edges (inversions), and thus immediately yields a successful alignment A between P and Q. Formally, the subsequence of P obtained by removing the positions V^* is also a subsequence of Q, and A is the monotone map matching them. Thus, $A(i) = \perp$ if and only if $i \in V^*$ and $\mathrm{cost}(A) = 2|E^*|$. Finally, using (A.1) we get that $m = |E^*| = \frac{1}{2}\,\mathrm{cost}(A) \ge \frac{1}{4}\,\mathrm{ed}(P,Q)$. ∎

We now give a claim that essentially lets us partition all potential alignments into a small number of groups of equivalent alignments.

Claim A.0.2. *Let P, Q be two permutations of length d. Fix $0 < \gamma < 1/2$ and a subset $S \subseteq [d]$. For an alignment A of P and Q (not necessarily successful), let $A_{|S} : S \to [d] \cup \{\perp\}$ be a function that is equal to the function A on the domain S. Define*

$$F = \{A_{|S} \mid A \text{ is an alignment of } P \text{ and } Q \text{ with } \mathrm{cost}(A) \le \gamma|S|\}.$$

Then $|F| \le (3e/\gamma)^{2\gamma|S|}$.

Proof. Let us denote $s = |S|$. An alignment of P with Q of cost at most γs can be described as deleting exactly γs symbols from P and exactly γs symbols from Q. (We assume here for simplicity that γs is an integer; otherwise, we round it up and change constants accordingly.) Clearly, we can bound $|F|$ by the number of such alignments between P and Q, namely $|F| \le \binom{d}{\gamma s}\binom{d}{\gamma s}$, but we aim to get a bound that depends on $s = |S|$ and not on d, by more carefully counting restrictions $A_{|S}$.

An alignment A of P with Q of cost at most γs can be described as first deleting exactly γs characters from P and then inserting into the resulting string exactly γs characters. Observe that $A_{|S}$ is completely determined from the following information: (a) which positions

[2] Another proof may be obtained using the $O(1)$-approximation in [GJKK07, Theorem 3.3].

in S are deleted; (b) how many characters are deleted between every two successive indices in S; and (c) how many characters are inserted between every two successive indices in S. (When we say two successive indices in S, it should be interpreted to include also 0 and $d+1$ as indices in S, and in particular (b) describes also how many characters before the first index in S are deleted from P.) Indeed, for each $i \in S$, data (a) determines whether $A(i) = \perp$. If $A(i) \neq \perp$, then $A(i) = i - d_i + a_i$ where d_i is the total number of deletions among indices $1, \dots, i-1$, which can be determined from data (a) and (b), and a_i is the total number of insertions before position i, which can be determined from data (c).

It remains to upper bound the number of possible outcomes to data (a)–(c). Clearly, the outcomes for (a) and (b) together can be upper bounded by the number of outcomes of throwing γs indistinguishable balls into $2s+2$ bins (a bin per element in S which may get at most one ball, a bin per each interval between elements in S and one extra bin to account for case when the cost is strictly less than γs). This upper bound is equal to $\binom{2s+2+\gamma s}{\gamma s}$ possible outcomes. The outcomes of data (c) can be similarly upper bounded by $\binom{s+1+\gamma s}{\gamma s}$. Together, we obtain that

$$|F| \leq \binom{2s+2+\gamma s}{\gamma s}\binom{s+1+\gamma s}{\gamma s} \leq \Big(\frac{e(2+2\gamma)}{\gamma}\Big)^{\gamma s}\Big(\frac{e(1+2\gamma)}{\gamma}\Big)^{\gamma s} \leq \Big(\frac{3e}{\gamma}\Big)^{2\gamma s},$$

which proves the claim. □

Having established the two claims, we proceed to prove Lemma 2.1.1, which states that with high probability, $\Omega(\mathrm{ed}(P,Q)) \leq \mathrm{ed}(\pi(P),\pi(Q)) \leq \mathrm{ed}(P,Q)$.

Proof of Lemma 2.1.1. Fix two permutations P and Q of length d. The inequality $\mathrm{ed}(\pi(P),\pi(Q)) \leq \mathrm{ed}(P,Q)$ follows immediately from the observation that every sequence of edit operations to transform P into Q can be applied also to transform $\pi(P)$ and $\pi(Q)$. It thus remains to prove the other direction. Assume for now that P and Q use the same symbols, i.e. $P([d]) = Q([d])$. We will later explain how the general case follows using a similar argument.

Apply Claim A.0.1 to P, Q, and extract $m \geq \mathrm{ed}(P,Q)/4$ inversions $\{i_1,j_1\},\dots,\{i_m,j_m\}$ such that $i_1,j_1,\dots,i_m,j_m$ are all distinct. Define $S = \{i_1,j_1,\dots,i_m,j_m\}$, hence $|S| = 2m$. Fix $\gamma = 1/100$ and let F be defined as in Claim A.0.2 (with respect to our P, Q, S and γ). By that claim, $|F| \leq (3e/\gamma)^{2\gamma|S|} = (3e/\gamma)^{4\gamma m}$. Note that F does not depend on π.

For every $f \in F$, let $\mathcal{E}_f$ be the event that all $i \in S$ with $f(i) \neq \perp$ satisfy $\pi(P(i)) = \pi(Q(f(i)))$. That is

$$\mathcal{E}_f = \bigwedge_{i \in S \setminus f^{-1}(\perp)} \{\pi(P(i)) = \pi(Q(f(i))\}.$$

We claim that

$$\Pr\Big[\mathrm{ed}(\pi(P),\pi(Q)) < \tfrac{1}{2}\gamma \cdot \mathrm{ed}(P,Q)\Big] \leq \Pr\Big[\bigcup_{f \in F} \mathcal{E}_f\Big]. \tag{A.2}$$

To prove the claim we show that $\mathrm{ed}(\pi(P),\pi(Q)) < \frac{1}{2}\gamma \cdot \mathrm{ed}(P,Q)$ implies that at least one of the events $\mathcal{E}_f$ happens. Indeed, suppose there is a successful alignment A between $\pi(P)$ and $\pi(Q)$ that has cost $\frac{1}{2}\gamma \cdot \mathrm{ed}(P,Q) \leq 2\gamma m = \gamma|S|$. Since A is successful, for all $i \in S \setminus A^{-1}(\perp)$, we must have $\pi(P(i)) = \pi(Q(A(i))$. Furthermore, we can think of A as an alignment between P and Q, and then by definition, its restriction $A_{|S}$ must be in F.

We now bound $\Pr[\mathcal{E}_f]$ for any fixed $f \in F$, i.e. $f = A_{|S}$ for some alignment A of cost at most $\gamma|S| = 2\gamma m$. Since S is the union of m inversions $\{i_t,j_t\}$ with distinct indices, for at least $(1-2\gamma)m$ of these inversions, we have that $f(i_t), f(j_t) \neq \perp$. For every such inversion $\{i_t, j_t\}$, it cannot be that both $P(i_t) = Q(f(i_t))$ and $P(j_t) = Q(f(j_t))$ (as that would contradict the fact that the alignment A is increasing). Let $a_t \neq b_t$ denote these two differing symbols (i.e. either $a_t = P(i_t)$, $b_t = Q(f(i_t))$ or $a_t = P(j_t)$, $b_t = Q(f(j_t))$), the event $\mathcal{E}_f$ can only occur if $\pi(a_t) = \pi(b_t)$. We thus obtain $(1-2\gamma)m$ requirements of the form $\pi(a_t) = \pi(b_t)$. These requirements have distinct symbols a_t in their left-hand sides (since they come from distinct positions in P), and similarly, the right-hand sides contain distinct symbols b_t. Altogether, every symbol in Σ may appear in at most two requirements, and thus we can extract (say greedily) a subcollection containing at least one half of these requirements, namely, at least $(1-2\gamma)m/2 \geq m/4$ requirements, such that every symbol appears in at most one requirement. Since π is a random function, the probability that all these requirements are satisfied is at most $2^{-m/4}$, and we conclude that $\Pr[\mathcal{E}_f] \leq 2^{-m/4}$.

To complete the proof of the lemma, we plug the last bound into (A.2) and use a union bound and Claim A.0.2, which altogether gives

$$\Pr\left[\mathrm{ed}(\pi(P), \pi(Q)) < \tfrac{1}{2}\gamma \cdot \mathrm{ed}(P, Q)\right] \leq (3e/\gamma)^{4\gamma m} \cdot 2^{-m/4} \leq 2^{-m/8}.$$

Finally, we extend the proof to the case where P and Q differ on some symbols, i.e., there is at least a symbol in P that is not in Q (and vice-versa). Define $\Sigma' = P([d]) \cap Q([d])$ to be the set of symbols that appear in both P and Q. Let P' be the string obtained by deleting from P the symbols not in Σ', and let Q' be obtained similarly from Q. Clearly, P' and Q' are permutations, they have the same length $d' = |\Sigma'|$, and they use exactly the same symbols. Furthermore, $\mathrm{ed}(P, Q) = \mathrm{ed}(P', Q') + \Theta(d - d')$. Applying Claim A.0.1 to P', Q', we get $m \geq \mathrm{ed}(P', Q')/4$ inversions $\{i'_1, j'_1\}, \ldots, \{i'_m, j'_m\}$ such that $i'_1, j'_1, \ldots, i'_m, j'_m$ are all distinct. Translating these positions to P yields m inversions $\{i_1, j_1\}, \ldots, \{i_m, j_m\}$ between P and Q, such that $i_1, j_1, \ldots, i_m, j_m$ are all distinct. We then let S contain the indices in these inversions and also the $d - d'$ indices in P containing the symbols not in Σ'. It is not difficult to see that we will still get $|S| \geq \Omega(\mathrm{ed}(P, Q))$. Inversions will give rise to requirements of the form $\pi(a) = \pi(b)$ as before, and each index i where $P(i) \notin \Sigma'$ gives rise to a requirement $\pi(P(i)) = \pi(Q(f(i)))$. Altogether, after removing indices i such that $f(i) = \perp$, we still get at least $|S|/8$ requirements whose variables $\pi(a), \pi(b)$ are all distinct. $\square$

Bibliography

[AB96] Ofer Amrani and Yair Be'ery. Efficient bounded-distance decoding of the hexacode and associated decoders for the Leech lattice and the Golay code. *IEEE Transactions on Communications*, 44:534–537, May 1996.

[ABV+94] Ofer Amrani, Yair Be'ery, Alexander Vardy, Feng-Wen Sun, and Henk C. A. van Tilborg. The Leech lattice and the Golay code: Bounded-distance decoding and multilevel constructions. *IEEE Transactions on Information Theory*, 40:1030–1043, July 1994.

[AC06] Nir Ailon and Bernard Chazelle. Approximate nearest neighbors and the Fast Johnson-Lindenstrauss Transform. *Proceedings of the Symposium on Theory of Computing (STOC)*, 2006.

[ACCL07] Nir Ailon, Bernard Chazelle, Seshadhri Comandur, and Ding Liu. Estimating the distance to a monotone function. *Random Structures and Algorithms*, 31:371–383, 2007. Previously appeared in RANDOM'04.

[ACP08] Alexandr Andoni, Dorian Croitoru, and Mihai Pătraşcu. Hardness of nearest neighbor under L-infinity. In *Proceedings of the Symposium on Foundations of Computer Science (FOCS)*, pages 424–433, 2008.

[AD99] David Aldous and Persi Diaconis. Longest increasing subsequences: from patience sorting to the Baik-Deift-Johansson theorem. *Bull. Amer. Math. Soc. (N.S.)*, 36(4):413–432, 1999.

[ADG+03] Alexandr Andoni, Michel Deza, Anupam Gupta, Piotr Indyk, and Sofya Raskhodnikova. Lower bounds for embedding edit distance into normed spaces. In *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 523–526, 2003.

[ADI+06] Alexandr Andoni, Mayur Datar, Nicole Immorlica, Piotr Indyk, and Vahab Mirrokni. Locality-sensitive hashing scheme based on p-stable distributions. *Nearest Neighbor Methods for Learning and Vision: Theory and Practice, Neural Processing Information Series, MIT Press*, 2006.

[ADI08] Alexandr Andoni, Khanh Do Ba, and Piotr Indyk. Block heavy hitters. *MIT Technical Report MIT-CSAIL-TR-2008-024*, 2008.

[ADIW09] Alexandr Andoni, Khanh Do Ba, Piotr Indyk, and David Woodruff. Efficient sketches for Earth-Mover Distance, with applications. In *Proceedings of the Symposium on Foundations of Computer Science (FOCS)*, 2009. To appear.

[AHL01] Helmut Alt and Laura Heinrich-Litan. Exact L_∞ nearest neighbor search in high dimensions. In *Proceedings of the ACM Symposium on Computational Geometry (SoCG)*, pages 157–163, 2001.

[AI05] Alexandr Andoni and Piotr Indyk. E2LSH: Exact Euclidean Locality-Sensitive Hashing. 2005. Implementation available at `http://web.mit.edu/andoni/www/LSH/index.html`.

[AI06a] Alexandr Andoni and Piotr Indyk. Efficient algorithms for substring near neighbor problem. In *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1203–1212, 2006.

[AI06b] Alexandr Andoni and Piotr Indyk. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. In *Proceedings of the Symposium on Foundations of Computer Science (FOCS)*, pages 459–468, 2006.

[AI08a] Alexandr Andoni and Piotr Indyk. Dimension reduction in kernel spaces from Locality-Sensitive Hashing. 2008. Manuscript.

[AI08b] Alexandr Andoni and Piotr Indyk. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. *Communications of the ACM*, 51(1):117–122, 2008.

[AIK08] Alexandr Andoni, Piotr Indyk, and Robert Krauthgamer. Earth mover distance over high-dimensional spaces. In *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 343–352, 2008.

[AIK09] Alexandr Andoni, Piotr Indyk, and Robert Krauthgamer. Overcoming the ℓ_1 non-embeddability barrier: Algorithms for product metrics. In *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 865–874, 2009.

[AIP06] Alexandr Andoni, Piotr Indyk, and Mihai Pătraşcu. On the optimality of the dimensionality reduction method. In *Proceedings of the Symposium on Foundations of Computer Science (FOCS)*, pages 449–458, 2006.

[AJKS02] Miklós Ajtai, T. S. Jayram, Ravi Kumar, and D. Sivakumar. Approximate counting of inversions in a data stream. In *Proceedings of the Symposium on Theory of Computing (STOC)*, pages 370–379, 2002.

[AJP10] Alexandr Andoni, T.S. Jayram, and Mihai Pătraşcu. Lower bounds for edit distance and product metrics via Poincaré-type inequalities. *Accepted to ACM-SIAM Symposium on Discrete Algorithms (SODA'10)*, 2010.

[AK07] Alexandr Andoni and Robert Krauthgamer. The computational hardness of estimating edit distance. In *Proceedings of the Symposium on Foundations of Computer Science (FOCS)*, pages 724–734, 2007. Accepted to *SIAM Journal on Computing* (FOCS'07 special issue).

[AK08a] Alexandr Andoni and Robert Krauthgamer. Distance estimation protocols of general metrics. *Manuscript*, 2008.

[AK08b] Alexandr Andoni and Robert Krauthgamer. The smoothed complexity of edit distance. In *Proceedings of International Colloquium on Automata, Languages and Programming (ICALP)*, pages 357–369, 2008.

[Aka] Akamai homepage: `http://www.akamai.com/html/technology/dataviz3.html`. Last accessed on August 30, 2009.

[ALN05] Sanjeev Arora, James R. Lee, and Assaf Naor. Euclidean distortion and the sparsest cut. In *Proceedings of the Symposium on Theory of Computing (STOC)*, pages 553–562, 2005.

[AMN+98] Sunil Arya, David M. Mount, Nathan S. Netanyahu, Ruth Silverman, and Angela Y. Wu. An optimal algorithm for approximate nearest neighbor searching. *J. ACM*, 6(45):891–923, 1998. Previously appeared in SODA'04.

[AMS99] Noga Alon, Yossi Matias, and Mario Szegedy. The space complexity of approximating the frequency moments. *J. Comp. Sys. Sci.*, 58:137–147, 1999. Previously appeared in STOC'96.

[AMS01] Dimitris Achlioptas, Frank McSherry, and Bernhard Schölkopf. Sampling techniques for kernel methods. In S. B. Thomas, G. Dietterich, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems (NIPS)*, 2001.

[AN10] Alexandr Andoni and Huy L. Nguyen. Near-tight bounds for testing Ulam distance. *Accepted to ACM-SIAM Symposium on Discrete Algorithms (SODA'10)*, 2010.

[And05] Alexandr Andoni. Approximate nearest neighbor problem in high dimensions. *M.Eng. Thesis, Massachusetts Institute of Technology*, June 2005.

[AO09] Alexandr Andoni and Krzysztof Onak. Approximating edit distance in near-linear time. In *Proceedings of the Symposium on Theory of Computing (STOC)*, pages 199–204, 2009.

[AR98] Yonatan Aumann and Yuval Rabani. An $O(\log k)$ approximate min-cut max-flow theorem and approximation algorithm. *SIAM J. Comput.*, 27(1):291–301 (electronic), 1998.

[ARV04] Sanjeev Arora, Satish Rao, and Umesh Vazirani. Expander flows, geometric embeddings, and graph partitionings. In *Proceedings of the Symposium on Theory of Computing (STOC)*, pages 222–231, 2004.

[AV06] Rosa I. Arriaga and Santosh Vempala. An algorithmic theory of learning: Robust concepts and random projection. *Machine Learning*, 62(2):161–182, 2006. Previously appeared in FOCS'99.

[BBD+02] Brian Babcock, Shivnath Babu, Mayur Datar, Rajeev Motwani, and Jennifer Widom. Models and issues in data stream systems. In *Proceedings of the ACM Symposium on Principles of Database Systems (PODS)*, pages 1–16, 2002.

[BBV06] Maria-Florina Balcan, Avrim Blum, and Santosh Vempala. Kernels as features: On kernels, margins, and low-dimensional mappings. *Machine Learning*, 65(1):79 – 94, October 2006.

[BC03] Bo Brinkman and Moses Charikar. On the impossibility of dimension reduction in ℓ_1. In *Proceedings of the Symposium on Foundations of Computer Science (FOCS)*, 2003.

[BC09] Joshua Brody and Amit Chakrabarti. A multi-round communication lower bound for gap Hamming and some consequences. In *Proceedings of the IEEE Conference on Computational Complexity*, 2009.

[BCFM00] Andrei Broder, Moses Charikar, Alan M. Frieze, and Michael Mitzenmacher. Min-wise independent permutations. *Journal of Computer Systems Sciences*, 60(3):630–659, 2000.

[BEK$^+$03] Tuğkan Batu, Funda Ergün, Joe Kilian, Avner Magen, Sofya Raskhodnikova, Ronitt Rubinfeld, and Rahul Sami. A sublinear algorithm for weakly approximating edit distance. In *Proceedings of the Symposium on Theory of Computing (STOC)*, pages 316–324, 2003.

[Ben75] John L. Bentley. Multidimensional binary search trees used for associative searching. *Communications of the ACM*, 18:509–517, 1975.

[BES06] Tuğkan Batu, Funda Ergün, and Cenk Sahinalp. Oblivious string embeddings and edit distance approximations. In *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 792–801, 2006.

[BFC08] Philip Bille and Martin Farach-Colton. Fast and compact regular expression matching. *Theoretical Computer Science*, 409(28):486–496, 2008.

[BGMZ97] Andrei Broder, Steve Glassman, Mark Manasse, and Geoffrey Zweig. Syntactic clustering of the web. *Proceedings of the Sixth International World Wide Web Conference*, pages 391–404, 1997.

[BJKK04] Ziv Bar-Yossef, T. S. Jayram, Robert Krauthgamer, and Ravi Kumar. Approximating edit distance efficiently. In *Proceedings of the Symposium on Foundations of Computer Science (FOCS)*, pages 550–559, 2004.

[BJKS04] Ziv Bar-Yossef, T. S. Jayram, Ravi Kumar, and D. Sivakumar. An information statistics approach to data stream and communication complexity. *J. Comput. Syst. Sci.*, 68(4):702–732, 2004.

[Blu06] Avrim Blum. Random projection, margins, kernels, and feature-selection. *LNCS 3940*, pages 52–68, 2006. Survey article based on an invited talk given at the 2005 PASCAL Workshop on Subspace, Latent Structure and Feature selection techniques.

[BOR99] Allan Borodin, Rafail Ostrovsky, and Yuval Rabani. Lower bounds for high dimensional nearest neighbor search and related problems. *Proceedings of the Symposium on Theory of Computing*, 1999.

[Bou85] Jean Bourgain. On Lipschitz embedding of finite metric spaces into Hilbert space. *Israel Journal of Mathematics*, 52:46–52, 1985.

[Bou02] Jean Bourgain. On the distributions of the Fourier spectrum of Boolean functions. *Israel J. Math.*, 131:269–276, 2002.

[BR02] Omer Barkol and Yuval Rabani. Tighter bounds for nearest neighbor search and related problems in the cell probe model. *J. Comput. Syst. Sci.*, 64(4):873–896, 2002. Previously appeared in STOC'00.

[Bro97] Andrei Broder. On the resemblance and containment of documents. *Proceedings of Compression and Complexity of Sequences*, pages 21–29, 1997.

[CCG+98] Moses Charikar, Chandra Chekuri, Ashish Goel, Sudipto Guha, and Serge Plotkin. Approximating a finite metric by a small number of tree metrics. *Proceedings of the Symposium on Foundations of Computer Science (FOCS)*, 1998.

[CCGL99] Amit Chakrabarti, Bernard Chazelle, Benjamin Gum, and Alexey Lvov. A lower bound on the complexity of approximate nearest-neighbor searching on the Hamming cube. *Proceedings of the Symposium on Theory of Computing (STOC)*, 1999.

[Cha02] Moses Charikar. Similarity estimation techniques from rounding. In *Proceedings of the Symposium on Theory of Computing (STOC)*, pages 380–388, 2002.

[CK06] Moses Charikar and Robert Krauthgamer. Embedding the Ulam metric into ℓ_1. *Theory of Computing*, 2(11):207–224, 2006.

[CKN09] Jeff Cheeger, Bruce Kleiner, and Assaf Naor. A $(\log n)^{\Omega(1)}$ integrality gap for the Sparsest Cut SDP. In *Proceedings of the Symposium on Foundations of Computer Science (FOCS)*, 2009. To appear.

[Cla88] Ken Clarkson. A randomized algorithm for closest-point queries. *SIAM Journal on Computing*, 17:830–847, 1988.

[CLRS01] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms*. MIT Press, 2nd edition, 2001.

[CM05a] Graham Cormode and S. Muthukrishnan. An improved data stream summary: the count-min sketch and its applications. *J. Algorithms*, 55(1):58–75, 2005.

[CM05b] Graham Cormode and S. Muthukrishnan. Space efficient mining of multigraph streams. In *Proceedings of the ACM Symposium on Principles of Database Systems (PODS)*, 2005.

[CM07] Graham Cormode and S. Muthukrishnan. The string edit distance matching problem with moves. *ACM Trans. Algorithms*, 3(1), 2007. Special issue on SODA'02.

[CNBYM01] Edgar Chávez, Gonzalo Navarro, Rricardo Baeza-Yates, and José L. Marroquin. Searching in metric spaces. *ACM Computing Surveys*, 33(3):273–321, September 2001.

[Cor03] Graham Cormode. *Sequence Distance Embeddings*. Ph.D. Thesis, University of Warwick. 2003.

[CPSV00] Graham Cormode, Mike Paterson, Suleyman Cenk Sahinalp, and Uzi Vishkin. Communication complexity of document exchange. In *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 197–206, 2000.

[CR93] Andrea Califano and Isidore Rigoutsos. FLASH: a fast look-up algorithm for string homology. In *International Conference on Intelligent Systems for Molecular Biology*, pages 56–64, 1993.

[CR04] Amit Chakrabarti and Oded Regev. An optimal randomised cell probe lower bounds for approximate nearest neighbor searching. *Proceedings of the Symposium on Foundations of Computer Science (FOCS)*, 2004.

[CS86] John H. Conway and Neil J. A. Sloane. Soft decoding techniques for codes and lattices, including the Golay code and the Leech lattice. *IEEE Trans. Inf. Theor.*, 32(1):41–50, 1986.

[CS93] John H. Conway and Neil J. A. Sloane. *Sphere Packings, Lattices, and Groups*. Springer-Verlag, New York, 1993.

[CS02] Moses Charikar and Amit Sahai. Dimension reduction in the ℓ_1 norm. In *Proceedings of the Symposium on Foundations of Computer Science (FOCS)*, pages 551–560, 2002.

[CSWY01] Amit Chakrabarti, Yaoyun Shi, Anthony Wirth, and Andrew C-C. Yao. Informational complexity and the direct sum problem for simultaneous message complexity. In *Proceedings of the Symposium on Foundations of Computer Science (FOCS)*, pages 270–278, 2001.

[DF87] Persi Diaconis and David Freedman. A dozen de finetti-style results in search of a theory. *Probabilités et Statistiques*, 23(2):397–423, 1987.

[DG99] Sanjoy Dasgupta and Anupam Gupta. An elementary proof of the Johnson–Lindenstrauss lemma. *ICSI technical report TR-99-006, Berkeley, CA*, 1999.

[Dia88] Persi Diaconis. Group representations in probability and statistics. *Lecture Notes–Monograph Series*, 11, 1988.

[DIIM04] Mayur Datar, Nicole Immorlica, Piotr Indyk, and Vahab Mirrokni. Locality-sensitive hashing scheme based on p-stable distributions. *Proceedings of the ACM Symposium on Computational Geometry (SoCG)*, 2004.

[DKSV06] Nikhil R. Devanur, Subhash A. Khot, Rishi Saket, and Nisheeth K. Vishnoi. Integrality gaps for sparsest cut and minimum linear arrangement problems. In *Proceedings of the Symposium on Theory of Computing (STOC)*, pages 537–546, 2006.

[DL97] Michel M. Deza and Monique Laurent. *Geometry of cuts and metrics*. Springer-Verlag, Berlin, 1997.

[DM03] Dennis DeCoste and Dominic Mazzoni. Fast query-optimized kernel machine via incremental approximate nearest support vectors. In *IEEE International Conference on Machine Learning*, 2003.

[Dub08] Moshe Dubiner. Bucketing coding and information theory for the statistical high dimensional nearest neighbor problem. *CoRR*, abs/0810.4182, 2008.

[dvOS97] Mark de Berg, Marc van Kreveld, Mark H. Overmars, and Otfried Schwarzkopf. *Computational Geometry: Algorithms and Applications*. Springer-Verlag, 1997.

[EKK+00] Funda Ergün, Sampath Kannan, Ravi Kumar, Ronitt Rubinfeld, and Manesh Viswanathan. Spot-checkers. *J. Comput. Syst. Sci.*, 60(3):717–751, 2000.

[Fac] Facebook statistics: `http://www.facebook.com/press/info.php?statistics`. Last accessed on August 30, 2009.

[Fag96] Ronald Fagin. Combining fuzzy information from multiple systems. *Proceedings of the ACM Symposium on Principles of Database Systems*, pages 216–227, 1996.

[Fag98] Ronald Fagin. Fuzzy queries in multimedia database systems (invited paper). *Proceedings of the ACM Symposium on Principles of Database Systems*, 1998.

[FCI99] Martin Farach-Colton and Piotr Indyk. Approximate nearest neighbor algorithms for hausdorff metrics via embeddings. *Proceedings of the Symposium on Foundations of Computer Science (FOCS)*, 1999.

[Fli] Flickr blog: `http://blog.flickr.net/en/2008/11/03/3-billion/`. Last accessed on August 30, 2009.

[FM85] Philippe Flajolet and G. Nigel Martin. Probabilistic counting algorithms for data base applications. *J. Comput. Syst. Sci.*, 31(2):182–209, 1985.

[FS02] Uriel Feige and Gideon Schechtman. On the optimality of the random hyperplane rounding technique for max cut. *Random Struct. Algorithms*, 20(3):403–440, 2002.

[GD04] Kristen Grauman and Trevor Darrell. Fast contour matching using approximate Earth Mover's Distance. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 220–227, 2004.

[GD05a] Kristen Grauman and Trevor Darrell. Efficient image matching with distributions of local invariant features. In *Proceedings IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 627–634, 2005.

[GD05b] Kristen Grauman and Trevor Darrell. The pyramid match kernel: Discriminative classification with sets of image features. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Beijing, China, October 2005.

[GD06] Kristen Grauman and Trevor Darrell. Approximate correspondences in high dimensions. In *Proceedings of Advances in Neural Information Processing Systems (NIPS)*, 2006.

[GIM99] Aristides Gionis, Piotr Indyk, and Rajeev Motwani. Similarity search in high dimensions via hashing. *Proceedings of the 25th International Conference on Very Large Data Bases (VLDB)*, 1999.

[GIM08] Sudipto Guha, Piotr Indyk, and Andrew McGregor. Sketching information divergences. *Journal of Machine Learning*, 72(1–2):5–19, 2008. Previously appeared in COLT'07.

[GJKK07] Parikshit Gopalan, T. S. Jayram, Robert Krauthgamer, and Ravi Kumar. Estimating the sortedness of a data stream. In *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 318–327, 2007.

[GPY94] Daniel H. Greene, Michal Parnas, and F. Frances Yao. Multi-index hashing for information retrieval. In *Proceedings of the Symposium on Foundations of Computer Science (FOCS)*, pages 722–731, 1994.

[Gus97] Dan Gusfield. *Algorithms on strings, trees, and sequences.* Cambridge University Press, Cambridge, 1997.

[GW95] Michel X. Goemans and David P. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *Journal of the ACM*, 42:1115–1145, 1995.

[HP01] Sariel Har-Peled. A replacement for voronoi diagrams of near linear size. In *Proceedings of the Symposium on Foundations of Computer Science (FOCS)*, pages 94–103, 2001.

[IM98] Piotr Indyk and Rajeev Motwani. Approximate nearest neighbor: towards removing the curse of dimensionality. *Proceedings of the Symposium on Theory of Computing (STOC)*, pages 604–613, 1998.

[IM03] Piotr Indyk and Jiří Matoušek. Discrete metric spaces. *CRC Handbook of Discrete and Computational Geometry, 2nd edition*, 2003.

[Ind98] Piotr Indyk. On approximate nearest neighbors in ℓ_∞ norm. *J. Comput. Syst. Sci., to appear.*, 1998. Preliminary version appeared in FOCS'98.

[Ind01a] Piotr Indyk. Algorithmic aspects of geometric embeddings (tutorial). In *Proceedings of the Symposium on Foundations of Computer Science (FOCS)*, pages 10–33, 2001.

[Ind01b] Piotr Indyk. Approximate algorithms for high-dimensional geometric problems. Invited talk at DIMACS Workshop on Computational Geometry'02. `http://people.csail.mit.edu/indyk/high.ps`, 2001.

[Ind01c] Piotr Indyk. *High-dimensional computational geometry.* Ph.D. Thesis. Department of Computer Science, Stanford University, 2001.

[Ind02a] Piotr Indyk. Approximate algorithms for high-dimensional geometric problems. *Invited talk at DIMACS Workshop on Computational Geometry. Available at* `http://theory.csail.mit.edu/~indyk/high.ps`, 2002.

[Ind02b] Piotr Indyk. Approximate nearest neighbor algorithms for Frechet metric via product metrics. *Proceedings of the ACM Symposium on Computational Geometry (SoCG)*, pages 102–106, 2002.

[Ind03a] Piotr Indyk. Better algorithms for high-dimensional proximity problems via asymmetric embeddings. *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2003.

[Ind03b] Piotr Indyk. Nearest neighbor in high dimensional spaces. *CRC Handbook of Discrete and Computational Geometry, 2nd edition*, 2003.

[Ind03c] Piotr Indyk. Nearest neighbors in high-dimensional spaces. *CRC Handbook of Discrete and Computational Geometry*, 2003.

[Ind04] Piotr Indyk. Approximate nearest neighbor under edit distance via product metrics. In *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 646–650, 2004.

[Ind06] Piotr Indyk. Stable distributions, pseudorandom generators, embeddings and data stream computation. *J. ACM*, 53(3):307–323, 2006. Previously appeared in FOCS'00.

[Ind07] Piotr Indyk. A near linear time constant factor approximation for euclidean bichromatic matching (cost). In *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2007.

[IT03] Piotr Indyk and Nitin Thaper. Fast color image retrieval via embeddings. *Workshop on Statistical and Computational Theories of Vision (at ICCV)*, 2003.

[IW03] Piotr Indyk and David Woodruff. Tight lower bounds for the distinct elements problem. *Proceedings of the Symposium on Foundations of Computer Science (FOCS)*, pages 283–290, 2003.

[IW05] Piotr Indyk and David Woodruff. Optimal approximations of the frequency moments of data streams. *Proceedings of the Symposium on Theory of Computing (STOC)*, 2005.

[JKKR04] T. S. Jayram, Subhash Khot, Ravi Kumar, and Yuval Rabani. Cell-probe lower bounds for the partial match problem. *Journal of Computer and Systems Sciences*, 69(3):435–447, 2004. See also STOC'03.

[JL84] William B. Johnson and Joram Lindenstrauss. Extensions of lipshitz mapping into hilbert space. *Contemporary Mathematics*, 26:189–206, 1984.

[JL01] William B. Johnson and Joram Lindenstrauss, editors. *Handbook of the geometry of Banach spaces. Vol. I.* North-Holland Publishing Co., Amsterdam, 2001.

[JS82] William B. Johnson and Gideon Schechtman. Embedding ℓ_p^m into ℓ_1^n. *Acta Mathematica*, 149:71–85, 1982.

[JW09] T.S. Jayram and David Woodruff. The data stream space complexity of cascaded norms. In *Proceedings of the Symposium on Foundations of Computer Science (FOCS)*, 2009. To appear.

[KKL88] Jeff Kahn, Gil Kalai, and Nathan Linial. The influence of variables on boolean functions. In *Proceedings of the Symposium on Foundations of Computer Science (FOCS)*, pages 68–80, 1988.

[KL04] Robert Krauthgamer and James R. Lee. Navigating nets: Simple algorithms for proximity search. *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2004.

[Kle97] Jon Kleinberg. Two algorithms for nearest-neighbor search in high dimensions. *Proceedings of the Twenty-Ninth Annual ACM Symposium on Theory of Computing*, 1997.

[KLMN04] Robert Krauthgamer, James R. Lee, Manor Mendel, and Assaf Naor. Measured descent: A new embedding method for finite metrics. *Geom. Funct. Anal.*, 2004.

[KN97] Eyal Kushilevitz and Noam Nisan. *Communication Complexity*. Cambridge University Press, 1997.

[KN06] Subhash Khot and Assaf Naor. Nonembeddability theorems via Fourier analysis. *Math. Ann.*, 334(4):821–852, 2006. Preliminary version appeared in FOCS'05.

[KNR99] Ilan Kremer, Noam Nisan, and Dana Ron. On randomized one-round communication complexity. *Computational Complexity*, 8(1):21–49, 1999.

[KOR00] Eyal Kushilevitz, Rafail Ostrovsky, and Yuval Rabani. Efficient search for approximate nearest neighbor in high dimensional spaces. *SIAM J. Comput.*, 30(2):457–474, 2000. Preliminary version appeared in STOC'98.

[KR06] Robert Krauthgamer and Yuval Rabani. Improved lower bounds for embeddings into L_1. In *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1010–1017, 2006.

[KV05] Subhash A. Khot and Nisheeth K. Vishnoi. The unique games conjecture, integrality gap for cut problems and embeddability of negative type metrics into ℓ_1. In *Proceedings of the Symposium on Foundations of Computer Science (FOCS)*, pages 53–62, 2005.

[KWZ95] Richard M. Karp, Orli Waarts, and Geoffrey Zweig. The bit vector intersection problem. In *Proceedings of the Symposium on Foundations of Computer Science (FOCS)*, pages 621–630, 1995.

[Lee67] John Leech. Notes on sphere packings. *Canadian Journal of Mathematics*, 1967.

[Lev65] Vladimir I. Levenshtein. Двоичные коды с исправлением выпадений, вставок и замещений символов. *Доклады Академии Наук СССР*, 4(163):845–848, 1965. Appeared in English as: V. I. Levenshtein, Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics Doklady* 10(8), 707–710, 1966.

[Lin02] Nathan Linial. Finite metric spaces - combinatorics, geometry and algorithms. *Proceedings of the International Congress of Mathematicians III*, pages 573–586, 2002.

[Liu04] Ding Liu. A strong lower bound for approximate nearest neighbor searching in the cell probe model. *Information Processing Letters*, 92:23–29, 2004.

[LLR94] Nathan Linial, Eran London, and Yuri Rabinovich. The geometry of graphs and some of its algorithmic applications. In *Proceedings of the Symposium on Foundations of Computer Science (FOCS)*, pages 577–591, 1994.

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

[LMS98] Gad M. Landau, Eugene W. Myers, and Jeanette P. Schmidt. Incremental string comparison. *SIAM J. Comput.*, 27(2):557–582, 1998.

[LN04] James Lee and Assaf Naor. Embedding the diamond graph in L_p and dimension reduction in L_1. *Geometric and Functional Analysis (GAFA)*, 14(4):745–747, 2004.

[LSP06] Svetlana Lazebnik, Cordelia Schmid, and Jean Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2006.

[Mar95] John I. Marden. *Analyzing and Modeling Rank Data*. Monographs on Statistics and Applied Probability 64. CRC Press, 1995.

[Mat96] Jiří Matoušek. On the distortion required for embedding finite metric spaces into normed spaces. *Israel Journal of Mathematics*, 93:333–344, 1996.

[Mat02] Jiří Matoušek. *Lectures on Discrete Geometry*. Springer, 2002.

[Mat07] Jiří Matoušek. Collection of open problems on low-distortion embeddings of finite metric spaces. March 2007. Available online. Last access in August, 2007.

[McG07] Andrew McGregor. *Processing Data Streams*. Ph.D. Thesis, University of Pennsylvania. 2007.

[Mei93] Stefan Meiser. Point location in arrangements of hyperplanes. *Information and Computation*, 106:286–303, 1993.

[MMR+01] Klaus-Robert Muller, Sebastian Mika, Gunnar Rätsch, Koji Tsuda, and Bernhard Schölkopf. An introduction to kernel-based learning algorithms. *IEEE Transactions on Neural Networks*, 12(2):181–201, March 2001.

[MNP06] Rajeev Motwani, Assaf Naor, and Rina Panigrahy. Lower bounds on locality sensitive hashing. In *Proceedings of the ACM Symposium on Computational Geometry (SoCG)*, pages 253–262, 2006.

[MNSW98] Peter B. Miltersen, Noam Nisan, Shmuel Safra, and Avi Wigderson. Data structures and asymmetric communication complexity. *Journal of Computer and System Sciences*, 1998.

[MP80] William J. Masek and Mike Paterson. A faster algorithm computing string edit distances. *J. Comput. Syst. Sci.*, 20(1):18–31, 1980.

[MR95] Rajeev Motwani and Prabhakar Raghavan. *Randomized Algorithms*. Cambridge University Press, 1995.

[MS00] S. Muthukrishnan and Cenk Sahinalp. Approximate nearest neighbors and sequence comparison with block operations. *Proceedings of the Symposium on Theory of Computing (STOC)*, pages 416–424, 2000.

[Mut03] S. Muthukrishnan. Data streams: Algorithms and applications (invited talk at soda'03). *Available at http://athos.rutgers.edu/∼muthu/stream-1-1.ps*, 2003.

[Mut09] S. Muthukrishnan. Data stream algorithms (notes from a series of lectures, with guest lecturer Andrew McGregor). *The 2009 Barbados Workshop on Computational Complexity*, March 2009.

[Nav01] Gonzalo Navarro. A guided tour to approximate string matching. *ACM Comput. Surv.*, 33(1):31–88, 2001.

[NCB] National Center for Biotechnology Information homepage: `http://www.ncbi.nlm.nih.gov/`. Last accessed on November 29, 2008.

[NS07] Assaf Naor and Gideon Schechtman. Planar earthmover is not in L_1. *SIAM Journal on Computing*, 37(3):804–826, 2007. An extended abstract appeared in FOCS'06.

[OR07] Rafail Ostrovsky and Yuval Rabani. Low distortion embedding for edit distance. *J. ACM*, 54(5), 2007. Preliminary version appeared in STOC'05.

[PA95] Janos Pach and Pankaj K. Agarwal. *Combinatorial Geometry*. John Wiley & Sons, NY, 1995.

[Pan06] Rina Panigrahy. Entropy-based nearest neighbor algorithm in high dimensions. *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2006.

[Păt08] Mihai Pătraşcu. (Data) STRUCTURES. In *Proceedings of the Symposium on Foundations of Computer Science (FOCS)*, 2008.

[Pis89] Gilles Pisier. *The volume of convex bodies and Banach space geometry*. Cambridge University Press, 1989.

[PRR95] Ramamohan Paturi, Sanguthevar Rajasekaran, and John Reif. The light bulb problem. *Information and Computation*, 117(2):187–192, 1995.

[PT06] Mihai Pătraşcu and Mikkel Thorup. Higher lower bounds for near-neighbor and further rich problems. *Proceedings of the Symposium on Foundations of Computer Science (FOCS)*, 2006.

[PTW08] Rina Panigrahy, Kunal Talwar, and Udi Wieder. A geometric approach to lower bounds for approximate near-neighbor search and partial match. In *Proceedings of the Symposium on Foundations of Computer Science (FOCS)*, pages 414–423, 2008.

[Rao99] Satish Rao. Small distortion and volume preserving embeddings for planar and Euclidean metrics. In *Proceedings of the 15th Annual Symposium on Computational Geometry*, pages 300–306, New York, 1999. ACM.

[RR07] Ali Rahimi and Benjamin Recht. Random features for large-scale kernel machines. In *Proceedings of Advances in Neural Information Processing Systems (NIPS)*, 2007.

[RTG00] Yossi Rubner, Carlo Tomasi, and Leonidas J. Guibas. The earth mover's distance as a metric for image retrieval. *International Journal of Computer Vision*, 40(2):99–121, 2000.

[Sah08] Süleyman Cenk Sahinalp. Edit distance under block operations. In Ming-Yang Kao, editor, *Encyclopedia of Algorithms*. Springer, 2008.

[Sam06] Hannan Samet. *Foundations of Multidimensional and Metric Data Structures*. Elsevier, 2006.

[Sch38] Issac J. Schoenberg. Metric spaces and positive definite functions. *Transactions of the American Mathematical Society*, 44(3):522–536, November 1938.

[SDI06] Gregory Shakhnarovich, Trevor Darrell, and Piotr Indyk, editors. *Nearest Neighbor Methods in Learning and Vision*. Neural Processing Information Series, MIT Press, 2006.

[SS02a] Michael Saks and Xiaodong Sun. Space lower bounds for distance approximation in the data stream model. In *Proceedings of the Symposium on Theory of Computing (STOC)*, pages 360–369, 2002.

[SS02b] Bernhard Schölkopf and Alexander J. Smola. *Learning with kernels. Support Vector Machines, Regularization, Optimization, and Beyond*. MIT University Press, Cambridge, 2002.

[Šte00] Daniel Štefankovič. Fourier transform in computer science. Master's thesis, University of Chicago, 2000.

[SU04] Cenk Sahinalp and Andrey Utis. Hardness of string similarity search and other indexing problems. In *Proceedings of International Colloquium on Automata, Languages and Programming (ICALP)*, pages 1080 – 1098, 2004.

[SV03] Amit Sahai and Salil Vadhan. A complete problem for statistical zero knowledge. *J. ACM*, 50(2):196–249, 2003.

[TT07] Tengo Terasawa and Yuzuru Tanaka. Spherical LSH for approximate nearest neighbor search on unit hypersphere. *Workshop on Algorithms and Data Structures*, 2007.

[Var95] Alexander Vardy. Even more efficient bounded-distance decoding of the hexacode, the Golay code, and the Leech lattice. *IEEE Transactions on Information Theory*, 41:1495–1499, September 1995.

[WC76] C. K. Wong and Ashok K. Chandra. Bounds for the string editing problem. *J. ACM*, 23(1):13–16, 1976.

[Woo04] David Woodruff. Optimal space lower bounds for all frequency moments. In *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 167–175, 2004.

[Woo07] David Woodruff. *Efficient and Private Distance Approximation in the Communication and Streaming Models*. PhD thesis, MIT, 2007.

[WSB98] Roger Weber, Hans J. Schek, and Stephen Blott. A quantitative analysis and performance study for similarity-search methods in high-dimensional spaces. *Proceedings of the 24th Int. Conf. Very Large Data Bases (VLDB)*, 1998.