SUBLINEAR ALGORITHMS FOR MASSIVE DATA COMPUTATIONS: STREAMING AND CORESETS

by

Samira Daruki

A dissertation submitted to the faculty of
The University of Utah
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

in

Computing

School of Computing
The University of Utah
May 2018

The University of Utah Graduate School

STATEMENT OF DISSERTATION APPROVAL

The dissertation of Samira Daruki

has been approved by the following supervisory committee members:

Suresh Venkatasubramanian, Chair(s) **05 Oct 2017**
Date Approved

Aditya Bhaskara, Member **16 Oct 2017**
Date Approved

Jeffrey Phillips, Member **05 Oct 2017**
Date Approved

Vivek Srikumar, Member **05 Oct 2017**
Date Approved

Justin Thaler, Member **05 Oct 2017**
Date Approved

by **Ross Whitaker**, Chair/Dean of

the Department/College/School of **Computing**

and by **David B. Kieda**, Dean of The Graduate School.

ABSTRACT

The contributions of this dissertation are centered around designing new algorithms in the general area of sublinear algorithms such as streaming, core sets and sublinear verification, with a special interest in problems arising from data analysis including data summarization, clustering, matrix problems and massive graphs.

In the first part, we focus on summaries and coresets, which are among the main techniques for designing sublinear algorithms for massive data sets. We initiate the study of coresets for uncertain data and study coresets for various types of range counting queries on uncertain data. We focus mainly on the indecisive model of locational uncertainty since it comes up frequently in real-world applications when multiple readings of the same object are made. In this model, each uncertain point has a probability density describing its location, defined as k distinct locations. Our goal is to construct a subset of the uncertain points, including their locational uncertainty, so that range counting queries can be answered by examining only this subset. For each type of query we provide coreset constructions with approximation-size trade-offs. We show that random sampling can be used to construct each type of coreset, and we also provide significantly improved bounds using discrepancy-based techniques on axis-aligned range queries.

In the second part, we focus on designing sublinear-space algorithms for approximate computations on massive graphs. In particular, we consider graph MAXCUT and correlation clustering problems and develop sampling based approaches to construct truly sublinear ($o(n)$) sized coresets for graphs that have polynomial (i.e., n^δ for any $\delta > 0$) average degree. Our technique is based on analyzing properties of random induced subprograms of the linear program formulations of the problems. We demonstrate this technique with two examples. Firstly, we present a sublinear sized core set to approximate the value of the MAX CUT in a graph to a $(1+\epsilon)$ factor. To the best of our knowledge, all the known methods in this regime rely crucially on near-regularity assumptions. Secondly, we apply the same framework to construct a sublinear-sized coreset for correlation clustering.

Our coreset construction also suggests 2-pass streaming algorithms for computing the MAX CUT and correlation clustering objective values which are left as future work at the time of writing this dissertation.

Finally, we focus on streaming verification algorithms as another model for designing sublinear algorithms. We give the first polylog space and sublinear (in number of edges) communication protocols for any streaming verification problems in graphs. We present efficient streaming interactive proofs that can verify maximum matching exactly. Our results cover all flavors of matchings (bipartite/ nonbipartite and weighted). In addition, we also present streaming verifiers for approximate metric TSP and exact triangle counting, as well as for graph primitives such as the number of connected components, bipartiteness, minimum spanning tree and connectivity. In particular, these are the first results for weighted matchings and for metric TSP in any streaming verification model. Our streaming verifiers use only polylogarithmic space while exchanging only polylogarithmic communication with the prover in addition to the output size of the relevant solution.

We also initiate a study of streaming interactive proofs (SIPs) for problems in data analysis and present efficient SIPs for some fundamental problems. We present protocols for clustering and shape fitting including minimum enclosing ball (MEB), width of a point set, k-centers and k-slab problem. We also present protocols for fundamental matrix analysis problems: We provide an improved protocol for rectangular matrix problems, which in turn can be used to verify k (approximate) eigenvectors of an $n \times n$ integer matrix A. In general our solutions use polylogarithmic rounds of communication and polylogarithmic total communication and verifier space.

This dissertation is dedicated to all the people in my life who have inspired me, making me think, believe, trust, help, learn, create, contribute, give back, fight and love.

CONTENTS

ACKNOWLEDGMENTS

First and foremost, I would like to thank my advisor, Suresh Venkatasubramanian for his wonderful guidance and support throughout my PhD. He has been a constant source of advice and ideas along this journey. He taught me what it means to do research and his enthusiasm and encouragement helped me to explore new research directions. Along with his extensive guidance, he also gave me this opportunity to collaborate with other researchers in the field which helped me to grow as a researcher.

I would also like to thank my thesis committee members, Jeff Phillips, Justin Thaler, Aditya Bhaskara and Vivek Srikumar.

I was fortunate to know Jeff from the first day in grad school and take his course on models of computations for massive data. That opened a new door to me as a fresh graduate student about new techniques for designing algorithms in modern world. He taught me about the power of sampling and how it can be used as a hammer for designing beautiful and simple algorithms. It is to him that I owe my interest in sublinear and streaming algorithms. I also started to work with him on my first research problem and his organized attitude towards research was a big inspiration for me as a young researcher.

Later on, I had the opportunity to visit Simons institute at University of California at Berkeley as a research scholar in theoretical foundations of big data program and had the pleasure of meeting Justin there. I was inspired by his line of work on streaming verification algorithms and tried to learn more about it which led to some nice collaborations and I am thankful for his guidance and his patience in replying to my emails.

I am also grateful to know Aditya later as a senior graduate student and had the opportunity to work with him. I thoroughly enjoyed the long hours we spent thinking and discussing which are among my best moments at graduate school. His intuitive way of attacking problems, confident attitude towards research and breadth of technical knowledge are the qualities which I admire.

Finally, I am thankful to interact with Vivek and have him on my research committee.

His friendly nature, ability and enthusiasm in understanding research problems out of his field is something I found remarkable. I got more excited about research in machine learning after interacting with him and he was always an encouragement for me to think more about the motivations and applied aspects of research problems. I hope to have more opportunities to interact with him in my future career.

I would also thank my mentors Kostas Tsioutsiouliklis and Stergios Stergiou at the Yahoo Research Labs, who gave me the opportunity to intern with their team which was a great learning experience. During my internship, I learned a lot about applied research in the industry, in different topics in Large Scale Data Mining and Machine Learning, which guided me through my career path and the transition from Academia to Industry.

I also thank all of my friends and colleagues at the Algorithms and Data Group at the University of Utah, whom I spent plenty of time with and learned about a diverse set of research topics.

Last but definitely not least, I want to express my deepest gratitude to my beloved parents, Ahmad and Mahin, and my dearest siblings, Saeed and Saeedeh. Without their consistent support and encouragement this journey was not possible. I hope this work makes them happy and proud.

CHAPTER 1

INTRODUCTION

The area of *sublinear algorithms* is a new rapidly emerging area of computer science. In recent years, there has been growing body of work on designing sublinear algorithms, which are algorithms that use space, time or communication that are sublinear in the input size, i.e., $o(\text{input size } n)$. This is mainly motivated by the increased interest in the design and analysis of algorithms for massive data sets that occur more frequently in various applications. Managing and analyzing such data sets requires reconsidering the traditional notions of efficient algorithms: the classic algorithmic models do not provide accurate means for modern large-scale applications and even linear cost algorithms can be too slow and expensive. Hence, there is the desire to develop algorithms whose complexities are not only polynomial, but in fact are sublinear in n. The range of questions that can be answered accurately using sublinear (or even polylogarithmic) space or time is enormous, and the underlying techniques of sketching, streaming, sampling and core sets have been proven to be a rich toolkit.

Constructing coresets is a technique for designing sublinear algorithms. This technique summarizes a large data set with a proxy set of potentially much smaller size that can guarantee error for certain classes of queries. Here the assumption is often that we have offline access to the data; however, due to the increasing size of the data this assumption may not be valid. This motivates study on the streaming algorithms, where data arrives in a streaming fashion and the goal is to extract only a small amount of information about the input (a "sketch") for computing the approximate answer to the query. However, it is shown that many problems (especially on graph data) are intractable in the streaming setting and require a prohibitive amount of space or number of passes over the data. This motivates considering a relaxation of the standard streaming model, called streaming interactive proofs (SIPs), in which a powerful third party (prover) assists with the computations to reduce the required memory while still verifying correctness by a sublinear amount of communication. Besides being practically motivated by outsourced computations, SIPs are closely related to Merlin-Arthur proofs with space-bounded verifiers.

In the following sections, we give an overview of the three main topics that this dissertation spans, and later we outline our results and contributions.

1.1 Summaries and Coresets

One of the most popular approaches to processing massive data is to first extract a compact representation (or synopsis) of the data and then perform further processing only on the representation itself, obtaining a good approximation to the original input. This approach significantly reduces the cost of processing, communicating and storing the data. Examples of this approach include techniques such as sampling, sketching and coresets. Coresets was first introduced in the field of computational geometry [4, 5, 40, 153]. Given a large data set P and a family of queries $\mathcal{A}$, then an *η-coreset* is a subset $S \subset P$ such that for all $r \in \mathcal{A}$ we obtain $\|r(P) - r(S)\| \leq \eta$ (note the notion of distance $\| \cdot \|$ between query results is problem specific). Initially used for smallest enclosing ball queries [40] and perhaps most famous in geometry for extent queries as η-kernels [4, 5], the coresets are now employed in many other problems such as clustering [27] and density estimation [153]. In recent years, methods for constructing coresets have become more mature theoretically and have been used to solve some well known open problems in computer science and machine learning. Coresets present a new approach to optimization in general and have huge success especially in tasks which use prohibitively large computation time or space. Furthermore, coresets offer a way to obtain algorithms that work in models such as distributed computing and the streaming model, where the space used by the algorithm must be significantly smaller than the input size: simply compute and store a coreset (typically, a coreset is much smaller than the input size), and then run a more expensive algorithm on the coreset rather than on the original input.

1.2 Streaming Graph Algorithms and Sketches

Large scale graphs are now a widely used tool for representing real world data. Many modern applications, such as search engines or social networks, require supporting various queries on large scale graphs efficiently. It is possible to store a massive graph on a large storage device, but the random access in these devices are often quite slow and the computation costs will be expensive.

To overcome the challenges which arise by computation on massive graphs, an important approach is maintaining a succinct representation that preserves certain properties of the graph (i.e., coreset). Another popular approach is to process such large graphs in the data stream model using a limited amount of space. The core task here is to construct a synopsis data structure which is easy to construct in streaming fashion while also yielding good approximation of the properties of the graph data. Many of these synopses are constructed using sketching (computing a linear projection of the data) and sampling techniques.

The techniques developed for graph streams are now finding application in other areas including data structures for dynamic graphs, approximation algorithms, and distributed and parallel computations. A nice survey of graph stream algorithms and related technical details can be found in [132].

1.3 Streaming Verification Algorithms

One of the main challenges in streaming computation on massive data is designing algorithms for potentially difficult problems in which computation or space requirements are prohibitive under the streaming model. In this case, we can consider outsourcing the storage and processing of the data stream to a more powerful third party, the cloud, but the data owner would like to be assured that the desired computation has been performed correctly. In this setting, the resource-limited verifier (data owner) sees a data stream and tries to solve the problem with the help of a more powerful prover (cloud) who sees the the same stream. This model can be viewed as a streaming modification of a classic interactive proof system (a streaming IP, or SIP), and has been the subject of a number of papers [45, 49, 64, 66, 70, 117, 119, 149] that have established sublinear (verifier) space and communication bounds for classic problems in streaming. Here the goal is to develop efficient (in terms of communication and space) interactive proof protocols for verifying computations which are streaming in nature and explore how interaction and communication can be helpful in attacking problems in data analysis as well as classic and fundamental problems in geometric and combinatorial algorithms and optimization under streaming inputs.

In this dissertation, we contribute to the study of sublinear algorithms for massive data computations in the following ways.

1.4 Chapter 2: Range Counting Coresets for Uncertain Data

There have been several works on computing coresets for different problems in geometry, data mining and machine learning, but here in this dissertation, we initiate the study of coresets for *uncertain* data. In recent years, uncertain data has become ubiquitous because of new technologies for collecting data which can only measure and collect the data in an imprecise way. As a result, there is a need for tools and techniques for mining and managing uncertain data and dealing with uncertainty, and exploring how the existing machinery for designing sublinear algorithms for massive data can be adapted to work efficiently in the uncertain case as well.

To model uncertainty on data, there are several formulations where each point $p \in P$ has an *independent* probability distribution μ_p describing its location and such a point is said to have *locational uncertainty*.

In Chapter 2, we study constructing coresets for various types of range counting queries on uncertain data. We focus mainly on the *indecisive* model of locational uncertainty since it comes up frequently in real-world applications when multiple readings of the same object are made. In this model, each uncertain point has a probability density describing its location, defined as k distinct locations. Our goal is to construct a subset of the uncertain points, including their locational uncertainty, so that range counting queries can be answered by just examining this subset. We study three distinct types of queries: RE queries return the expected number of points in a query range; RC queries return the number of points in the range with probability at least a threshold; RQ queries return the probability that fewer than some threshold fraction of the points are in the range. In both RC and RQ coresets the threshold is provided as part of the query.

We provide the first results for RE-, RC-, and RQ-coresets with guarantees. In particular we show that a random sample T of size $O((1/\varepsilon^2)(\nu + \log(1/\delta))$ with probability $1 - \delta$ is an ε-RC coreset for any family of ranges $\mathcal{A}$ whose associated range space has VC-dimension

ν. If we enforce that each uncertain point has k possible locations, then a sample T of size $O((1/\varepsilon^2)(\nu + \log(k/\delta))$ suffices for an ε-RE coreset.

Then we leverage discrepancy-based techniques [51, 131] for some specific families of ranges $\mathcal{A}$, to improve these bounds to $O((1/\varepsilon)\mathsf{poly}(k, \log(1/\varepsilon)))$. This is an important improvement since $1/\varepsilon$ can be quite large (say 100 or more), while k, interpreted as the number of readings of a data point, is small for many applications (say 5). In $\mathbb{R}^1$, for one-sided ranges we construct ε-RE and ε-RC coresets of size $O((\sqrt{k}/\varepsilon)\log(k/\varepsilon))$. For axis-aligned rectangles in $\mathbb{R}^d$ we construct ε-RE coresets of size $O((\sqrt{k}/\varepsilon) \cdot \log^{\frac{3d-1}{2}}(k/\varepsilon))$ and ε-RC coresets of size $O((k^{3d+\frac{1}{2}}/\varepsilon)\log^{6d-\frac{1}{2}}(k/\varepsilon)))$. Finally, for RQ queries, we show that to get useful bounds for approximating the answers via coresets, it is required to allow an α-error in the threshold associated with the query itself, in addition to the standard ϵ-error (for $0 < \epsilon \leq 1$) in the returned answer. We prove that any ε-RE coreset of size t is also an $(\varepsilon, \alpha_{\varepsilon,t})$-RQ coreset with value $\alpha_{\varepsilon,t} = \varepsilon + \sqrt{(1/2t)\ln(2/\varepsilon)}$.

These results leverage new connections between uncertain points and both discrepancy of permutations and colored range searching that may be of independent interest.

1.5 Chapter 3: Sublinear Algorithms for Maxcut and Correlation Clustering

When dealing with large *graphs*, the sublinear paradigm has proven to be effective in two ways. Firstly, for *dense* graphs, it has long been known that simple sampling strategies allow us to estimate good approximations even for NP-hard problems. In the streaming setting, viewing a graph as a large matrix and using sketching/sampling techniques has led to algorithms that use space that is sublinear in the number of edges, but not the number of vertices (the so-called *semi-streaming* paradigm).

Pushing beyond this limit has proven to be difficult. In one direction, the upper bounds based on sketching and (edge) sampling appear to require access to all the vertices in the graph. In another direction, there are now a number of lower bounds indicating that any single-pass streaming algorithm that uses strictly sublinear space cannot get a $(1+\epsilon)$-approximation (for MaxCut and matching, for example). Note that for MaxCut, it is trivial to get a 2-approximation in $\log m$ space by counting edges, and edge sampling techniques can get $(1+\epsilon)$ approximations in $\tilde{O}(n)$ space [1].

While the lower bounds suggest that the situation is hopeless, the actual constructions used in these bounds are very sparse (the graphs have $\Theta(n)$ edges). Given the success of sampling techniques for *dense* graphs as well as graphs with some kind of regularity assumption, the question we pose in this paper is: "can we get truly sublinear approximations for (some) graph problems in between (easy) dense graphs and (hard) sparse graphs?"

In Chapter 3, we develop sampling-based approaches to produce core sets of truly sublinear size ($O(n^{1-\delta})$) for graphs that have $\Omega(n^{1+\delta})$ edges (where $0 < \delta \leq 1$). Our approach draws on ideas first developed outside the realm of sublinear algorithms: the key insight is to write down a linear program expressing the relevant problem, and then analyze properties of an *induced* linear program formed by randomly selecting variables (i.e., vertices) from the linear program. We apply this paradigm to two graph problems that have been studied in the sublinear setting: MaxCut and correlation clustering. In both cases, our approach yields the aforementioned sublinear-sized core set. Our algorithm for constructing coresets suggests a 2-pass streaming algorithm for these two problems which is left as a future work.

In order to prove these results, we have to overcome two major obstacles. Firstly, results of Andoni et al. [18] suggest that maintaining *all* cuts in a graph approximately requires an $\Omega(n)$ size sketch, and most prior sublinear algorithms for computing MaxCut in fact maintain all cuts. Thus in order get below the $\Omega(n)$ barrier it seems important to focus only on the *maximum cut*. Our approach based on the linear programming formulation is quite different in this sense. Secondly, offline efficient PTASs for MaxCut use uniform sampling of the graph and rely on the variance in vertex degrees being small (either by enforcing a regularity condition [33, 77] or by assuming density [16], which ensures that most vertices have degree $\Omega(n)$). We avoid such assumptions by using a *biased* sampling strategy (i.e., vertices of higher degree need to be sampled with proportionally higher probabilities). Analyzing non-uniform vertex sampling and its effect on the values of optimization problems (MaxCut and correlation clustering) on large graphs is our main contribution.

[1] As usual, $\tilde{O}(f(n)) = O(f(n)\text{poly}\log n)$.

Our algorithm itself is very simple, and indeed the high level argument can be easily summarized (Section 3.3). The main complexity in our analysis is in showing a concentration bound for the behavior of random induced subprograms of linear programs (a slight variant of this was considered in earlier works such as [16]).

1.6 Chapter 5: Streaming Verification for Graph Properties

All prior work on streaming graph verification has been in the annotation model, which in practice resembles a 1-round SIP (a single message from prover to verifier after the stream has been read). In Chapter 5, we present streaming interactive proofs (SIPs) for graph problems that are traditionally hard for streaming, such as for the maximum matching problem (in bipartite and general graphs, both weighted and unweighted) as well for approximating the traveling salesperson problem. While the streaming model of computation has been extremely effective for processing numeric and matrix data, its ability to handle large graphs is limited, even in the so-called *semistreaming* model where the streaming algorithm is permitted to use space quasilinear in the number of vertices. Recent breakthroughs in graph sketching [132] have led to space-efficient approximations for many problems in the semistreaming model but canonical graph problems like matchings have been shown to be provably hard.

It is known [111] that no better than a $1 - 1/e$ approximation to the maximum cardinality matching is possible in the streaming model, even with space $\tilde{O}(n)$. It was also known that even allowing limited communication (effectively a single message from the prover) required a space-communication product of $\Omega(n^2)$ [45, 64]. Our results show that even allowing a few more rounds of communication dramatically improves the space-communication tradeoff for matching, as well as yielding *exact* verification. We note that streaming algorithms for matching vary greatly in performance and complexity depending in whether the graph is weighted or unweighted, bipartite or nonbipartite. In contrast, our results apply to all forms of matching. Interestingly, the special case of *perfect matching*, by virtue of being in RNC [115], admits an efficient SIP via results by Goldwasser, Kalai and Rothblum [89] and Cormode, Thaler and Yi [66]. Similarly for triangle counting, the best streaming algorithm [14] yields an additive εn^3 error estimate in polylogarithmic space, and again in the annotation model (effectively a single round of communication) the best result yields a space-communication product of $n^2 \log^2 n$, which is almost exponentially worse than the bound we obtain. We note that counting triangles is a classic problem in the sublinear algorithms literature, and identifying optimal space and communication bounds for this problem was posed as an open problem by Graham Cormode in the Bertinoro sublinear algorithms workshop [60]. Our bound for verifying a $3/2 + \epsilon$ approximation for the TSP in dynamic graphs is also interesting: a trivial 2-approximation in the semistreaming model follows via the MST, but it is open to improve this bound (even on a grid) [145].

In general, our results can be viewed as providing further insight into the tradeoff between space and communication in sublinear algorithms. The annotation model of verification provides $\Omega(n^2)$ lower bounds on the space-communication product for the problems we consider: in that light, the fact that we can obtain polynomially better bounds with only a constant number of rounds demonstrates the power of just a few rounds of interaction. We also note that virtually all of the canonical hard problems for streaming algorithms (Index [49], Disjointness [28, 31], Boolean Hidden Matching [43, 75, 121]) admit efficient SIPs. A SIP for Index was presented in [49] and we present SIPs for Disjointness and Boolean Hidden Matching here as well.

1.7 Chapter 6: Streaming Verification for Data Analysis

The shift from direct computation to outsourcing in the cloud has led to new ways of thinking about massive scale computation. In the *verification* setting, computational effort is split between a computationally weak client (the *verifier*) who owns the data and wants to solve a desired problem, and a more powerful server (the *prover*) which performs the computations. Here the client has only limited (streaming) access to the data, as well as a bounded ability to talk with the server (measured by the amount of communication), but wishes to verify the correctness of the prover's answers.

There are now many third party "cloud" services that can perform intensive computational tasks on large data. Examples include Amazon EC2 services, Microsoft's Azure cloud platform, Google Compute Engine and even a host of specialized platforms for large-scale data analysis.[2] These servers are not computationally limited: they typically comprise large clusters of computing nodes.

In Chapter 6 of this thesis, we initiate a study of streaming interactive proofs for problems in data analysis and present efficient SIPs for some fundamental problems. We present protocols for clustering and shape fitting as well as an improved protocol for rectangular matrix multiplication. In particular, We give *3-message* SIPs that can verify a minimum enclosing ball (MEB) and the width of a point set *exactly* with polylogarithmic space and communication costs. We present polylogarithmic round protocols with polylogarithmic communication and verifier space for verifying optimal k-centers and k-slabs in Euclidean space. We also show a simple 3-message protocol for verifying a 2-approximation to the k-center in a metric space, via simple adaptation of the Gonzalez 2-approximation for k-center. We also present protocols for fundamental matrix analysis problems: We provide an improved protocol for rectangular matrix problems, which in turn can be used to verify k (approximate) eigenvectors of an $n \times n$ integer matrix A. In general our solutions use polylogarithmic rounds of communication and polylogarithmic total communication and verifier space. For special cases (when optimality certificates can be verified easily), we present constant round protocols with similar costs. For rectangular matrix multiplication and eigenvector verification, our protocols work in the more restricted annotated data streaming model (which essentially corresponds to one-message SIPs), and use sublinear (but not polylogarithmic) communication.

Furthermore, in Chapter 4 we provide an overview of main techniques and tools for streaming verification algorithms which we use later in Chapter 5 and 6. We end this dissertation by presenting a summary of all the technical results achieved along with some open directions for future research work.

Main parts of this dissertation have been published as papers or manuscripts. Chapter 2 is published as [1], Chapter 5 is published as [2], Chapter 6 is published as [70] and Chapter 3 is a manuscript [36] submitted and was under review at the point this dissertation is written.

[2] See, for example, `http://aws.amazon.com/ec2/`, `http://azure.microsoft.com/en-us/`, `https://cloud.google.com/compute/`, `https://bigml.com/`, and `https://dato.com/index.html`

CHAPTER 2

RANGE COUNTING CORESETS FOR UNCERTAIN DATA

Techniques for constructing coresets are becoming more relevant in the era of big data; they summarize a large data set P with a proxy set S of potentially much smaller size that can guarantee error for certain classes of queries. They also shed light on the limits of how much information can possibly be represented in a small set of data.

2.1 Overview

The two main themes of this chapter are around coresets and uncertain data, which we give a short overview of them.

2.1.1 Coresets

In this chapter, we focus on a specific type of coreset called an η-sample [52, 98, 153] that can be thought of as preserving density queries and that has deep ties to the basis of learning theory [20]. Given a set of objects X (often $X \subset \mathbb{R}^d$ is a point set) and a family of subsets $\mathcal{A}$ of X, then the pair $(X, \mathcal{A})$ is called an *range space*. Often $\mathcal{A}$ are specified by containment in geometric shapes, for instance as all subsets of X defined by inclusion in any ball, any half space, or any axis-aligned rectangle. Now an η-sample of $(X, \mathcal{A})$ is a single subset $S \subset X$ such that

$$\max_{r \in \mathcal{A}} \left| \frac{|X \cap r|}{|X|} - \frac{|S \cap r|}{|S|} \right| \leq \eta.$$

For any query range $r \in \mathcal{A}$, subset S approximates the relative density of X in r with error at most η.

There are various works on constructing coresets for different problems. We give a brief overview of some of this related work here.

In [6], Agarwal et al. give an overview of coreset based algorithms for geometric approximations including extent measures and other related optimization problems. Reference [99] presents coresets for k-means and k-median clustering. In addition, [79] gives constant-size coresets for PCA and projective clustering. Reference [56] also studies coresets for k-median and k-means clustering in metric and euclidean spaces. Reference [26] presents optimal coresets for balls and [82] develops coresets for polytope distance. In [71], Dasgupta et al. provide sampling algorithms and coresets for ℓ_p regression problem and in [78] Feldman et al. study coresets and sketches for high-dimensional subspace approximation problems. There are several other works which study the construction of coresets with the goal of designing faster algorithms; a comprehensive review of this sizable literature is outside of the scope of this dissertation.

2.1.2 Uncertain Data

Another emerging notion in data analysis is modeling uncertainty in points. There are several formulations of these problems where each point $p \in P$ has an *independent* probability distribution μ_p describing its location and such a point is said to have *locational uncertainty*. In *imprecise points* (also called *deterministic uncertainty*) model, a data point $p \in P$ could be anywhere within a fixed continuous range and was originally used for analyzing precision errors. The worst case properties of a point set P under the imprecise model have been well-studied [29, 91, 92, 100, 124, 127, 135, 137, 152]. In *indecisive points* (or *attribute uncertainty* in database literature [144]) model, each $p_i \in P$ is able to take one of k distinct locations $\{p_{i,1}, p_{i,2}, \ldots, p_{i,k}\}$ with possibly different probabilities, modeling when multiple readings of the same object have been made [3, 8, 61, 62, 104, 151].

We also note another common model of *existential uncertainty* (similar to *tuple uncertainty* in database literature [144] but a bit less general) where the location or value of each $p \in P$ is fixed, but the point may not exist with some probability, modeling false readings [62, 108, 109, 144].

In this chapter, we will focus mainly on the indecisive model of locational uncertainty since it comes up frequently in real-world applications [8, 151] (when multiple readings of the same object are made, and typically k is small) and can be used to approximately represent more general continuous representations [103, 139].

2.2 Problem Statement

Combining the two notions coreset and uncertain data leads to the question: can we create a coreset (specifically for η-samples) of uncertain input data? A few more definitions are required to rigorously state this question. In fact, we develop three distinct notions of how to define the coreset error in uncertain points. One corresponds to range counting queries, another to querying the mean, and the third to querying the median (actually it approximates the rank for all quantiles).

For an uncertain point set $P = \{p_1, p_2, \ldots, p_n\}$ with each $p_i = \{p_{i,1}, p_{i,2}, \ldots p_{i,k}\} \subset \mathbb{R}^d$ we say that $Q \Subset P$ is a *transversal* if $Q \in p_1 \times p_2 \times \ldots \times p_n$. I.e., $Q = (q_1, q_2, \ldots, q_n)$ is an instantiation of the uncertain data P and can be treated as a "certain" point set, where each q_i corresponds to the location of p_i. $\mathbf{Pr}_{Q \Subset P}[\zeta(Q)]$, (resp. $\mathbf{E}_{Q \Subset P}[\zeta(Q)]$) represents the probability (resp. expected value) of an event $\zeta(Q)$ where Q is instantiated from P according to the probability distribution on the uncertainty in P.

As stated, our goal is to construct a subset of uncertain points $T \subset P$ (including the distribution of each point p's location, μ_p) that preserves specific properties over a family of subsets $(P, \mathcal{A})$. For completeness, the first variation we list cannot be accomplished purely with a coreset as it requires $\Omega(n)$ space.

- *Range Reporting (RR) Queries* support queries of a range $r \in \mathcal{A}$ and a threshold τ, and return all $p_i \in P$ such that $\mathbf{Pr}_{Q \Subset P}[q_i \in r] \geq \tau$. Note that the fate of each $p_i \in P$ depends on no other $p_j \in P$ where $i \neq j$, so they can be considered independently. Building indexes for this model have been studied [57, 69, 147, 156] and effectively solved in $\mathbb{R}^1$ [3].

- *Range Expectation (RE) Queries* consider a range $r \in \mathcal{A}$ and report the expected number of uncertain points in r, $\mathbf{E}_{Q \Subset P}[|r \cap Q|]$. The linearity of expectation allows summing the individual expectations each point $p \in P$ is in r. Single queries in this model have also been studied [41, 101, 102].

- *Range Counting (RC) Queries* support queries of a range $r \in \mathcal{A}$ and a threshold τ, but only return the number of $p_i \in P$ which satisfy $\mathbf{Pr}_{Q \Subset P}[q_i \in r] \geq \tau$. The effect of each $p_i \in P$ on the query is separate from that of any other $p_j \in P$ where $i \neq j$. A random sampling heuristic [155] has been suggested without proof of accuracy.

- *Range Quantile (RQ) Queries* take a query range $r \in \mathcal{A}$, and report the full cumulative density function on the number of points in the range $\mathbf{Pr}_{Q \Subset P}[|r \cap Q|]$. Thus for a query range r, this returned structure can produce for any value $\tau \in [0,1]$ the probability that τn or fewer points are in r. Since this is no longer an expectation, the linearity of expectation cannot be used to decompose this query along individual uncertain points.

Across all queries we consider, there are two main ways we can approximate the answers. The first and most standard way is to allow an ε-error (for $0 \leq \varepsilon \leq 1$) in the returned answer for RQ, RE, and RC. The second way is to allow an α-error in the *threshold* associated with the query itself. As will be shown, this is not necessary for RR, RE, or RC, but is required to get useful bounds for RQ. Finally, we will also consider probabilistic error δ, demarcating the probability of failure in a randomized algorithm (such as random sampling). We strive to achieve these approximation factors with a small size coreset $T \subset P$ as follows:

RE: For a given range r, let $r(Q) = |Q \cap r| / |Q|$, and let $E_{r(P)} = \mathbf{E}_{Q \Subset P}[r(Q)]$. $T \subset P$ is an *ε-RE coreset* of $(P, \mathcal{A})$ if for all queries $r \in \mathcal{A}$ we have $\left| E_{r(P)} - E_{r(T)} \right| \leq \varepsilon$.

RC: For a range $r \in \mathcal{A}$, let $G_{P,r}(\tau) = \frac{1}{|P|} \left| \{ p_i \in P \mid \mathbf{Pr}_{Q \Subset P}[q_i \in r] \geq \tau \} \right|$ be the fraction of points in P that are in r with probability at least some threshold τ. Then $T \subset P$ is an *ε-RC coreset* of $(P, \mathcal{A})$ if for all queries $r \in \mathcal{A}$ and all $\tau \in [0,1]$ we have $|G_{P,r}(\tau) - G_{T,r}(\tau)| \leq \varepsilon$.

RQ: For a range $r \in \mathcal{A}$, let $F_{P,r}(\tau) = \mathbf{Pr}_{Q \Subset P}[r(Q) \leq \tau] = \mathbf{Pr}_{Q \Subset P}\left[\frac{|Q \cap r|}{|Q|} \leq \tau \right]$ be the probability that at most a τ fraction of P is in r. Now $T \subset P$ is an *(ε, α)-RQ coreset* of $(P, \mathcal{A})$ if for all $r \in \mathcal{A}$ and $\tau \in [0,1]$ there exists a $\gamma \in [\tau - \alpha, \tau + \alpha]$ such that $|F_{P,r}(\tau) - F_{T,r}(\gamma)| \leq \varepsilon$. In such a situation, we also say that $F_{T,r}$ is an (ε, α)*-quantization* of $F_{P,r}$.

A natural question is whether we can construct a $(\varepsilon, 0)$-RQ coreset where there is not a secondary α-error term on τ. We demonstrate that there are no useful nontrivial bounds on the size of such a coreset.

When the (ε, α)-quantization $F_{T,r}$ need not be explicitly represented by a coreset T, then Löffler and Phillips [104, 126] show a different small space representation that can replace it in the above definition of an (ε, α)-RQ coreset with probability at least $1 - \delta$. First randomly create $m = O((1/\varepsilon^2) \log(1/\delta))$ transversals $Q_1, Q_2, \ldots, Q_m$, and for each transversal Q_i

create an α-sample S_i of $(Q_i, \mathcal{A})$. Then to satisfy the requirements of $F_{T,r}(\tau)$, there exists some $\gamma \in [\tau - \alpha, \tau + \alpha]$ such that we can return $(1/m)|\{S_i \mid r(S_i) \leq \gamma\}|$, and it will be within ε of $F_{P,r}(\tau)$. However, this is subverting the attempt to construct and understand a coreset to answer these questions. A coreset T (our goal) can be used as proxy for P as opposed to querying m distinct point sets. This alternate approach also does not shed light into how much information can be captured by a small size point set, which is provided by bounds on the size of a coreset.

2.2.1 Simple Example

We illustrate a simple example with $k = 2$ and $d = 1$, where $n = 10$ and the $nk = 20$ possible locations of the 10 uncertain points are laid out in order:

$$p_{1,1} < p_{2,1} < p_{3,1} < p_{4,1} < p_{5,1} < p_{6,1} < p_{3,2} < p_{7,1} < p_{8,1} < p_{8,2} < p_{9,1} < p_{10,1} < p_{5,2}$$
$$< p_{10,2} < p_{2,2} < p_{9,2} < p_{7,2} < p_{4,2} < p_{6,2} < p_{1,2}.$$

We consider a coreset $T \subset P$ that consists of the uncertain points $T = \{p_1, p_3, p_5, p_7, p_9\}$. Now consider a specific range $r \in \mathcal{I}_+$, a one-sided interval that contains $p_{5,2}$ and smaller points, but not $p_{10,2}$ and larger points. We can now see that $F_{T,r}$ is an $(\varepsilon' = 0.1016, \alpha = 0.1)$-quantization of $F_{P,r}$ in Figure 2.1; this follows since at $F_{P,r}(0.75) = 0.7734$ either $F_{T,r}(x)$ is at most 0.5 for $x \in [0.65, 0.8)$ and is at least 0.875 for $x \in [0.8, 0.85]$. Also observe that

$$\left| E_{r(P)} - E_{r(T)} \right| = \left| \frac{13}{20} - \frac{7}{10} \right| = \frac{1}{20} = \varepsilon.$$

When these errors (the (ε', α)-quantization and ε-error) hold for *all* ranges in some range space, then T is an (ε', α)-RQ coreset or ε-RC coreset, respectively.

To understand the error associated with an RC coreset, also consider the threshold $\tau = 2/3$ with respect to the range r. Then in range r, $2/10$ of the uncertain points from P are in r with probability at least $\tau = 2/3$ (points p_3 and p_8). Also $1/5$ of the uncertain points from T are in r with probability at least $\tau = 2/3$ (only point p_3). So there is 0 RC error for this range and threshold.

2.3 Discrepancy and Permutations

The key tool we will use to construct small coresets for uncertain data is discrepancy of range spaces, and specifically those defined on permutations. Consider a set X, a

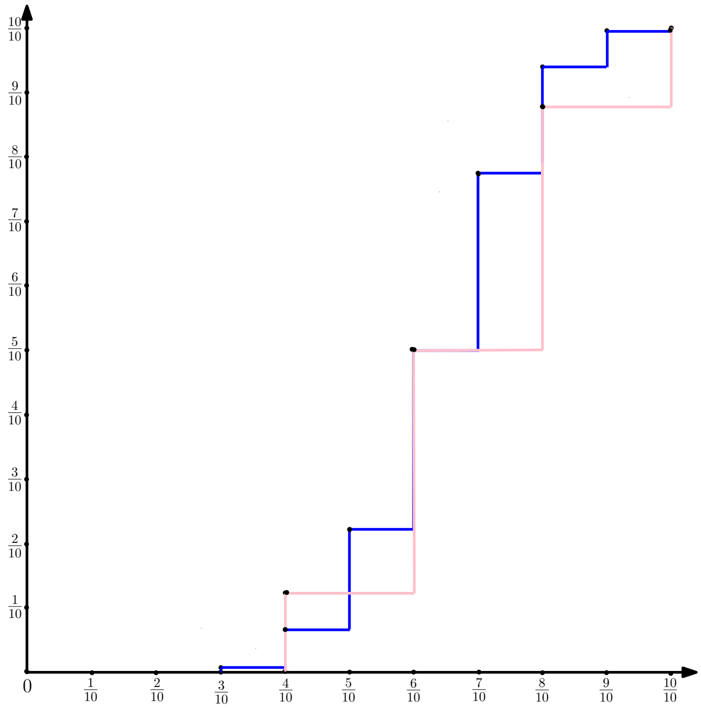


Figure 2.1: Example cumulative density functions ($F_{T,r}$, in red with fewer steps, and $F_{P,r}$, in blue with more steps) on uncertain point set P and a coreset T for a specific range.

range space $(X, \mathcal{A})$, and a coloring $\chi : X \to \{-1, +1\}$. Then for some range $A \in \mathcal{A}$, the discrepancy is defined $\mathsf{disc}_\chi(X, A) = |\sum_{x \in X \cap A} \chi(x)|$. We can then extend this to be over all ranges $\mathsf{disc}_\chi(X, \mathcal{A}) = \max_{A \in \mathcal{A}} \mathsf{disc}_\chi(X, A)$ and over all colorings $\mathsf{disc}(X, \mathcal{A}) = \min_\chi \mathsf{disc}_\chi(X, \mathcal{A})$.

Consider a ground set (P, Σ_k) where P is a set of n objects, and $\Sigma_k = \{\sigma_1, \sigma_2, \ldots, \sigma_k\}$ is a set of k permutations over P so each $\sigma_j : P \to [n]$. We can also consider a family of ranges $\mathcal{I}_k$ as a set of intervals defined on *one* of the k permutations so $I_{x,y,j} \in \mathcal{I}_k$ is defined so $P \cap I_{x,y,j} = \{p \in P \mid x < \sigma_j(p) \leq y\}$ for $x < y \in [0, n]$ and $j \in [k]$. The pair $((P, \Sigma_k), \mathcal{I}_k)$ is then a range space, defining a set of subsets of P.

A canonical way to obtain k permutations from an uncertain point set $P = \{p_1, p_2, \ldots, p_n\}$ is as follows. Define the *jth canonical traversal* of P as the set $P_j = \cup_{i=1}^n p_{i,j}$. When each $p_{i,j} \in \mathbb{R}^1$, the sorted order of each canonical traversal P_j defines a permutation on P as $\sigma_j(p_i) = |\{p_{i',j} \in P_j \mid p_{i',j} \leq p_{i,j}\}|$, that is $\sigma_j(p_i)$ describes how many locations (including $p_{i,j}$) in the traversal P_j have value less than or equal to $p_{i,j}$. In other words, σ_j describes the sorted order of the jth point among all uncertain points. Then, given an uncertain point set, let the canonical traversals define the *canonical k-permutation* as (P, Σ_k).

A geometric view of the permutation range space embeds P as n fixed points in $\mathbb{R}^k$ and considers ranges which are defined by inclusion in $(k-1)$-dimensional slabs, defined by two parallel half spaces with normals aligned along one of the coordinate axes. Specifically, the jth coordinate of the ith point is $\sigma_j(p_i)$, and if the range is on the jth permutation, then the slab is orthogonal to the jth coordinate axis.

Another useful construction from an uncertain point set P is the set P_{cert} of all locations any point in P might occur. Specifically, for every uncertain point set P we can define the corresponding certain point set $P_{\mathsf{cert}} = \bigcup_{i \in [n]} p_i = \bigcup_{j \in [k]} P_j = \bigcup_{i \in [n], j \in [k]} p_{i,j}$. We can also extend any coloring χ on P to a coloring in P_{cert} by letting $\chi_{\mathsf{cert}}(p_{ij}) = \chi(p_i)$, for $i \in [n]$ and $j \in [k]$. Now we can naturally define the discrepancy induced on P_{cert} by any coloring χ of P as $\mathsf{disc}_{\chi_{\mathsf{cert}}}(P_{\mathsf{cert}}, \mathcal{A}) = \max_{r \in \mathcal{A}} \sum_{p_{i,j} \in P_{\mathsf{cert}} \cap r} \chi(p_{i,j})$.

2.4 Low Discrepancy to ε-Coreset

There is a well-studied relationship between range spaces that admit low-discrepancy colorings, and creating ε-samples of those range spaces [34, 51, 52, 131]. Mainly in the 90s Chazelle and Matousek [51, 52, 54, 130, 131] led the development of the method to convert from a low-discrepancy coloring to a coreset that allowed for approximate range queries. The key relationship states that if $\mathsf{disc}(X, \mathcal{A}) = \gamma \log^{\omega}(n)$, then there exists an ε-sample of $(P, \mathcal{A})$ of size $O((\gamma/\varepsilon) \cdot \log^{\omega}(\gamma/\varepsilon))$ [139], for values γ, ω independent of n or ε. Construct the coloring, and with equal probability discard either all points colored either -1 or those colored $+1$. This roughly halves the point set size, and also implies zero over-count in expectation for any fixed range. Repeat this coloring and reduction of points until the desired size is achieved. This can be done efficiently in a distributed manner through a merge-reduce framework [52]. The take-away is that a method for a low-discrepancy coloring directly implies a method to create an ε-sample, where the counting error is in expectation zero for any fixed range. Here we describe and extend these results in much more detail and state a bit more specifically for our setting.

Theorem 2.1 (Phillips [139, 140]). *Consider a point set P of size n and a family of subsets $\mathcal{A}$. Assume an $O(n^{\beta})$ time algorithm to construct a coloring $\chi : P \to \{-1, +1\}$ so $\mathsf{disc}_{\chi}(P, \mathcal{A}) = O(\gamma \log^{\omega} n)$ where β, γ, and ω are constant algorithm parameters dependent on $\mathcal{A}$, but not P (or n).*

There exists an algorithm to construct an ε-sample of $(P, \mathcal{A})$ of size $g(\varepsilon, \mathcal{A}) = O((\gamma/\varepsilon) \log^{\omega}(\gamma/\varepsilon))$ in time $O(n \cdot g(\varepsilon, \mathcal{A})^{\beta-1})$.

Note that we ignored nonexponential dependence on ω and β since in our setting they are data and problem independent constants. But we are more careful with γ terms since they depend on k, the number of locations of each uncertain point.

We restate the algorithm and analysis here for completeness, using $g = g(\varepsilon, \mathcal{A})$ for shorthand. Divide P into n/g parts $\{\bar{P}_1, \bar{P}_2, \ldots, \bar{P}_{n/g}\}$ of size $k = 4(\beta+2)g$. Assume this divides evenly and n/g is a power of two; otherwise pad P and adjust g by a constant. Until there is a single set, repeat the following two stages. In stage 1, for $\beta + 2$ steps, pair up all remaining sets, and for all pairs (e.g. P_i and P_j) construct a low-discrepancy coloring χ on $P_i \cup P_j$ and discard all points colored -1 (or $+1$ at random). In the $(\beta+3)$rd step pair up all sets, but do not construct a coloring and halve. That is every epoch ($\beta + 3$ steps) the size of remaining sets double, otherwise they remain the same size. When a single set remains, stage 2 begins; it performs the color-halve part of the above procedure until $\mathsf{disc}(P, \mathcal{A}) \leq \varepsilon n$ as desired.

We begin analyzing the error on a single coloring.

Lemma 2.1. *The set $P^+ = \{p \in P \mid \chi(p) = +1\}$ is an $(\mathsf{disc}_{\chi}(P, \mathcal{A})/n)$-sample of $(P, \mathcal{A})$.*

Proof.

$$\max_{R \in \mathcal{A}} \left| \frac{|P \cap R|}{|P|} - \frac{|P^+ \cap R|}{|P^+|} \right| = \max_{R \in \mathcal{A}} \left| \frac{|P \cap R| - 2|P^+ \cap R|}{n} \right| \leq \frac{\mathsf{disc}_{\chi}(P, \mathcal{A})}{n}.$$

□

We also note two simple facts [51, 131]:

(S1) If Q_1 is an ε-sample of P_1 and Q_2 is an ε-sample of P_2, then $Q_1 \cup Q_2$ is an ε-sample of $P_1 \cup P_2$.

(S2) If Q is an ε_1-sample of P and S is an ε_2 sample of Q, then S is an $(\varepsilon_1 + \varepsilon_2)$-sample of P.

Note that (S1) (along with Lemma 2.1) implies the arbitrarily decomposing P into n/g sets and constructing colorings of each achieves the same error bound as doing so on just one. And (S2) implies that chaining together rounds adds the error in each round. It follows that if we ignore the $(\beta + 3)$rd step in each epoch, then there is 1 set remaining after $\log(n/g)$ steps. The error caused by each step is $\mathsf{disc}(g, \mathcal{A})/g$ so the total error is $\log(n/g)(\gamma \log^{\omega} g)/g = \varepsilon$. Solving for g yields $g = O(\frac{\gamma}{\varepsilon} \log(\frac{n\varepsilon}{\gamma}) \log^{\omega}(\frac{\gamma}{\varepsilon}))$.

Thus to achieve the result stated in the theorem the $(\beta + 3)$rd step skip of a reduce needs to remove the $\log(n\varepsilon/\gamma)$ term from the error. This works! After $\beta + 3$ steps, the size of each set is $2g$ and the discrepancy error is $\gamma \log^{\omega}(2g)/2g$. This is just more than half of what it was before, so the total error is now:

$$\sum_{i=0}^{\frac{\log(n/g)}{\beta+3}} (\beta + 3)\gamma \log^{\omega}(2^i g)/(2^i g) = \Theta(\beta(\gamma \log^{\omega} g)/g) = \varepsilon.$$

Solving for g yields $g = O(\frac{\beta\gamma}{\varepsilon} \log^{\omega}(1/\varepsilon))$ as desired. Stage 2 can be shown not to asymptotically increase the error.

To achieve the runtime we again start with the form of the algorithm without the half-skip on every $(\beta + 3)$rd step. Then the first step takes $O((n/g) \cdot g^{\beta})$ time. And each ith step takes $O((n/2^{i-1})g^{\beta-1})$ time. Since each subsequent step takes half as much time, the runtime is dominated by the first $O(ng^{\beta-1})$ time step.

For the full algorithm, the first epoch ($\beta + 3$ steps, including a skipped halve) takes $O(ng^{\beta-1})$ time, and the ith epoch takes $O(n/2^{(\beta+2)i}(g2^i)^{\beta-1}) = O(ng^{\beta-1}/2^{3i})$ time. Thus the time is still dominated by the first epoch. Again, stage 2 can be shown not to affect this runtime, and the total runtime bound is achieved as desired, and completes the proof.

Finally, we state a useful corollary about the expected error being 0. This holds specifically when we choose to discard the set P^+ or $P^- = \{p \in P \mid \chi(p) = -1\}$ at random on each halving.

Corollary 2.1. *The expected error for any range $R \in \mathcal{A}$ on the ε-sample T created by Theorem 2.1 is*

$$\mathbf{E}\left[\frac{|R \cap P|}{|P|} - \frac{|T \cap R|}{|T|}\right] = 0.$$

Note that there is no absolute value taken inside $\mathbf{E}[\cdot]$, so technically this measures the expected undercount.

2.4.1 RE-Discrepancy

We are also interested in achieving these same results for RE-discrepancy. To this end, the algorithms are identical. Lemma 2.4 replaces Lemma 2.1. (S1) and (S2) still hold. Nothing else about the analysis depends on properties of disc or RE-disc, so Theorem 2.1 can be restated for RE-discrepancy.

Theorem 2.2. *Consider an uncertain point set P of size n and a family of subsets $\mathcal{A}$ of P_{cert}. Assume an $O(n^\beta)$ time algorithm to construct a coloring $\chi : P \to \{-1, +1\}$ so $\textsf{RE-disc}_\chi(P, \mathcal{A}) = O(\gamma \log^\omega n)$ where β, γ, and ω are constant algorithm parameters dependent on $\mathcal{A}$, but not P (or n). There exists an algorithm to construct an ε-RE coreset of $(P, \mathcal{A})$ of size $g(\varepsilon, \mathcal{A}) = O((\gamma/\varepsilon) \log^\omega(\gamma/\varepsilon))$ in time $O(n \cdot g(\varepsilon, \mathcal{A})^{\beta-1})$.*

2.5 RE Coresets

First we will analyze ε-RE coresets through the P_{cert} interpretation of uncertain point set P. The canonical transversals P_j of P will also be useful. In Section 2.5.2 we will relate these results to a form of discrepancy.

Lemma 2.2. *$T \subset P$ is an ε-RE coreset for $(P, \mathcal{A})$ if and only if $T_{cert} \subset P_{\mathsf{cert}}$ is an ε-sample for $(P_{\mathsf{cert}}, \mathcal{A})$.*

Proof. First note that since $\mathbf{Pr}[p_i = p_{ij}] = \frac{1}{k}\ \forall i, j$, hence by linearity of expectations we have that $\mathbf{E}_{Q \Subset P}[|Q \cap r|] = \sum_{i=1}^n E[|p_i \cap r|] = \frac{1}{k}|P_{\mathsf{cert}} \cap r|$. Now, direct computation gives us:

$$\left| \frac{|P_{\mathsf{cert}} \cap r|}{|P_{\mathsf{cert}}|} - \frac{|T_{\mathsf{cert}} \cap r|}{|T_{\mathsf{cert}}|} \right| = \left| \frac{|P_{\mathsf{cert}} \cap r|}{k|P|} - \frac{|T_{\mathsf{cert}} \cap r|}{k|T|} \right| = \left| E_{r(P)} - E_{r(T)} \right| < \varepsilon.$$

□

The next implication enables us to determine an ε-RE coreset on P from ε-samples on each $P_j \Subset P$. Recall P_j is the jth canonical transversal of P for $j \in [k]$, and is defined similarly for a subset $T \subset P$ as T_j.

Lemma 2.3. *Given a range space $(P_{\mathsf{cert}}, \mathcal{A})$, if we have $T \subset P$ such that T_j is an ε-sample for $(P_j, \mathcal{A})$ for all $j \in [k]$, then T is an ε-RE coreset for $(P, \mathcal{A})$.*

Proof. Consider an arbitrary range $r \in \mathcal{R}$, and compute directly $\left| E_{r(P)} - E_{r(T)} \right|$. Recalling that $E_{r(P)} = \frac{|P_{\mathsf{cert}} \cap r|}{|P_{\mathsf{cert}}|}$ and observing that $|P_{\mathsf{cert}}| = k|P|$, we get that:

$$\left| E_{r(P)} - E_{r(T)} \right| = \left| \frac{\sum_{j=1}^k |P_j \cap r|}{k|P|} - \frac{\sum_{j=1}^k |T_j \cap r|}{k|T|} \right| \leq \frac{1}{k} \sum_{j=1}^k \left| \frac{|P_j \cap r|}{|P|} - \frac{|T_j \cap r|}{|T|} \right| \leq \frac{1}{k}(k\varepsilon) = \varepsilon.$$

□

2.5.1 Random Sampling

We show that a simple random sampling gives us an ε-RE coreset of P.

Theorem 2.3. *For an uncertain points set P and range space $(P_{\mathsf{cert}}, \mathcal{A})$ with VC-dimension ν, a random sample $T \subset P$ of size $O((1/\varepsilon^2)(\nu + \log(k/\delta)))$ is an ε-RE coreset of $(P, \mathcal{I})$ with probability at least $1 - \delta$.*

Proof. A random sample T_j of size $O((1/\varepsilon^2)(\nu + \log(1/\delta')))$ is an ε-sample of any $(P_j, \mathcal{A})$ with probability at least $1 - \delta'$ [125]. Now assuming $T \subset P$ resulted from a random sample on P, it induces the k disjoint canonical transversals T_j on T, such that $T_j \subset P_j$ and $|T_j| = O((1/\varepsilon^2)(\nu + \log(1/\delta')))$ for $j \in [k]$. Each T_j is an ε-sample of $(P_j, \mathcal{A})$ for any single $j \in [k]$ with probability at least $1 - \delta'$. Following Lemma 2.3 and using union bound, we conclude that $T \subset P$ is an ε-RE coreset for uncertain point set P with probability at least $1 - k\delta'$. Setting $\delta' = \delta/k$ proves the theorem. □

2.5.2 RE-Discrepancy and its Properties

Next we extend the well-studied relationship between geometric discrepancy and ε-samples on certain data towards ε-RE coresets on uncertain data.

We first require precise and slightly nonstandard definitions.

We introduce a new type of discrepancy based on the expected value of uncertain points called *RE-discrepancy*. Let P_χ^+ and P_χ^- denote the sets of uncertain points from P colored $+1$ or -1, respectively, by χ. Then $\mathsf{RE\text{-}disc}_\chi(P, r) = |P| \cdot |E_{r(P_\chi^+)} - E_{r(P)}|$ for any $r \in \mathcal{A}$. The usual extensions then follow: $\mathsf{RE\text{-}disc}_\chi(P, \mathcal{A}) = \max_{r \in \mathcal{A}} \mathsf{RE\text{-}disc}(P, r)$ and $\mathsf{RE\text{-}disc}(P, \mathcal{A}) = \min_\chi \mathsf{RE\text{-}disc}_\chi(P, \mathcal{A})$. Note that $(P, \mathcal{A})$ is technically not a range space, since $\mathcal{A}$ defines subsets of P_{cert} in this case, not of P.

Lemma 2.4. *Consider a coloring $\chi : P \to \{-1, +1\}$ such that $\mathsf{RE\text{-}disc}_\chi(P, \mathcal{A}) = \gamma \log^\omega(n)$ and $|P_\chi^+| = n/2$. Then the set P_χ^+ is an ε-RE coreset of $(P, \mathcal{A})$ with $\varepsilon = \frac{\gamma}{n} \log(n)$.*

Furthermore, if a subset $T \subset P$ has size $n/2$ and is an $(\frac{\gamma}{n} \log^\omega(n))$-RE coreset, then it defines a coloring χ (where $\chi(p_i) = +1$ for $p_i \in T$) that has $\mathsf{RE\text{-}disc}_\chi(P, \mathcal{A}) = \gamma \log^\omega(n)$.

Proof. We prove the second statement, the first follows symmetrically. We refer to the subset T as P_χ^+. Let $r = \arg\max_{r' \in \mathcal{A}} |E_{r'(P)} - E_{r'(P_\chi^+)}|$. This implies $\frac{\gamma}{n} \log^\omega n \geq |E_{r(P)} - E_{r(P_\chi^+)}| =$

$\frac{1}{n}\mathsf{RE\text{-}disc}_{\chi}(P, r)$. □

We can now recast RE-discrepancy to discrepancy on P_{cert}. From Lemma 2.2 we know that $\left|\frac{|P_{\mathsf{cert}} \cap r|}{k|P|} - \frac{|T_{\mathsf{cert}} \cap r|}{k|T|}\right| = \left|E_{r(P)} - E_{r(T)}\right|$ and after some basic substitutions we obtain the following.

Lemma 2.5. *RE-disc*$_{\chi}(P, \mathcal{A}) = \frac{1}{k}$*disc*$_{\chi_{cert}}(P_{cert}, \mathcal{A})$.

This does not immediately solve ε-RE coresets by standard discrepancy techniques on P_{cert} because we need to find a coloring χ on P. A coloring χ_{cert} on P_{cert} may not be consistent across all $p_{i,j} \in p_i$. The following lemma allows us to reduce this to a problem of coloring each canonical transversal P_j.

Lemma 2.6. *RE-disc*$_{\chi}(P, \mathcal{A}) \leq \max_j$ *disc*$_{\chi_{cert}}(P_j, \mathcal{A})$.

Proof. For any $r \in \mathcal{A}$ and any coloring χ (and the corresponding χ_{cert}), we can write P as a union of disjoint transversals P_j to obtain

$$\begin{aligned}\mathsf{disc}_{\chi_{\mathsf{cert}}}(P_{\mathsf{cert}}, r) &= \left|\sum_{j=1}^{k} \sum_{p_{ij} \in P_j \cap r} \chi_{\mathsf{cert}}(p_{ij})\right| \leq \sum_{j=1}^{k} \left|\sum_{p_{ij} \in P_j \cap r} \chi_{\mathsf{cert}}(p_{ij})\right| \\ &\leq \sum_{j=1}^{k} \mathsf{disc}_{\chi_{\mathsf{cert}}}(P_j, r) \leq k \max_j \mathsf{disc}_{\chi_{\mathsf{cert}}}(P_j, r).\end{aligned}$$

Since this holds for every $r \in \mathcal{A}$, hence (using Lemma 2.5)

$$\mathsf{RE\text{-}disc}_{\chi}(P, \mathcal{A}) = \frac{1}{k}\mathsf{disc}_{\chi_{\mathsf{cert}}}(P_{\mathsf{cert}}, \mathcal{A}) \leq \max_j \mathsf{disc}_{\chi_{\mathsf{cert}}}(P_j, \mathcal{A}).$$ □

2.5.3 ε-RE Coresets in $\mathbb{R}^1$

Lemma 2.7. *Consider uncertain point set P with $P_{cert} \subset \mathbb{R}^1$ and the range space $(P_{cert}, \mathcal{I}_+)$ with ranges defined by one-sided intervals of the form $(-\infty, x]$, then RE-disc$(P, \mathcal{I}) = O(\sqrt{k}\log n)$.*

Proof. Spencer et al. [146] show that $\mathsf{disc}((P, \Sigma_k), \mathcal{I}_k)$ is $O(\sqrt{k}\log n)$. Since we obtain the Σ_k from the canonical transversals P_1 through P_k, by definition this results in upper bounds on the the discrepancy over all P_j (it bounds the max). Lemma 2.6 then gives us the bound on $\mathsf{RE\text{-}disc}(P, \mathcal{I})$. □

As we discussed in Section 2.4 the low RE-discrepancy coloring can be iterated in a merge-reduce framework as developed by Chazelle and Matousek [52]. With Theorem 2.2 we can prove the following theorem.

Theorem 2.4. *Consider uncertain point set P and range space $(P_{cert}, \mathcal{I}_+)$ with ranges defined by one-sided intervals of the form $(-\infty, x]$, then an ε-RE coreset can be constructed of size $O((\sqrt{k}/\varepsilon)\log(k/\varepsilon))$.*

Since expected value is linear, we have

$$\mathsf{RE\text{-}disc}_\chi(P,(-\infty,x]) - \mathsf{RE\text{-}disc}_\chi(P,(-\infty,y)) = \mathsf{RE\text{-}disc}_\chi(P,[y,x])$$

for $y < x$ and the above result also holds for the family of two-sided ranges $\mathcal{I}$.

2.5.4 ε-RE Coresets for Rectangles in $\mathbb{R}^d$

Here let P be a set of n uncertain points where each possible location of a point $p_{i,j} \in \mathbb{R}^d$. We consider a range space $(P_{\mathsf{cert}}, \mathcal{R}_d)$ defined by d-dimensional axis-aligned rectangles.

Each canonical transversal P_j for $j \in [k]$ no longer implies a unique permutation on the points (for $d > 1$). But, for any rectangle $r \in \mathcal{R}$, we can represent any $r \cap P_j$ as the disjoint union of points P_j contained in intervals on a predefined set of $(1+\log n)^{d-1}$ permutations [37]. Spencer et al. [146] showed there exists a coloring χ such that

$$\max_j \mathsf{disc}_\chi(P_j, \mathcal{R}) = O(D_\ell(n)\log^{d-1} n),$$

where $\ell = (1+\log n)^{d-1}$ is the number of defined permutations and $D_\ell(n)$ is the discrepancy of ℓ permutations over n points and ranges defined as intervals on each permutation. Furthermore, they showed $D_\ell(n) = O(\sqrt{\ell}\log n)$.

To get the RE-discrepancy bound for $P_{\mathsf{cert}} = \cup_{j=1}^k P_j$, we first decompose P_{cert} into the k point sets P_j of size n. We then obtain $(1+\log n)^{d-1}$ permutations over points in each P_j, and hence obtain a family Σ_ℓ of $\ell = k(1+\log n)^{d-1}$ permutations over all P_j. $D_\ell(n) = O(\sqrt{\ell}\log n)$ yields

$$\mathsf{disc}((P,\Sigma_\ell),\mathcal{I}_\ell) = O(\sqrt{k}\log^{\frac{d+1}{2}} n).$$

Now each set $P_j \cap r$ for $r \in \mathcal{R}_d$, can be written as the disjoint union of $O(\log^{d-1} n)$ intervals of Σ_ℓ. Summing up over each interval, we get that $\mathsf{disc}(P_j, \mathcal{R}) = O(\sqrt{k}\log^{\frac{3d-1}{2}} n)$ for each

j. By Lemma 2.6 this bounds the RE-discrepancy as well. Finally, we can again apply the merge-reduce framework of Chazelle and Matousek [52] (via Theorem 2.2) to achieve an ε-RE coreset.

Theorem 2.5. *Consider uncertain point set P and range space $(P_{\mathsf{cert}}, \mathcal{R}_d)$ (for $d > 1$) with ranges defined by axis-aligned rectangles in $\mathbb{R}^d$. Then an ε-RE coreset can be constructed of size $O((\sqrt{k}/\varepsilon)\log^{\frac{3d-1}{2}}(k/\varepsilon))$.*

2.6 RC Coresets

Recall that an ε-RC coreset T of a set P of n uncertain points satisfies that for all queries $r \in \mathcal{A}$ and all thresholds $\tau \in [0,1]$ we have $|G_{P,r}(\tau) - G_{T,r}(\tau)| \leq \varepsilon$, where $G_{P,r}(\tau)$ represents the fraction of points from P that are in range r with probability at least τ.

In this setting, given a range $r \in \mathcal{A}$ and a threshold $\tau \in [0,1]$ we can let the pair $(r,\tau) \in \mathcal{A} \times [0,1]$ define a range $R_{r,\tau}$ such that each $p_i \in P$ is either in or not in $R_{r,\tau}$. Let $(P, \mathcal{A} \times [0,1])$ denote this range space. If $(P_{\mathsf{cert}}, \mathcal{A})$ has VC-dimension ν, then $(P, \mathcal{A} \times [0,1])$ has VC-dimension $O(\nu + 1)$; This is according to the Corollary 5.23 in [98]. This implies that random sampling works to construct ε-RC coresets.

Theorem 2.6. *For uncertain point set P and range space $(P_{\mathsf{cert}}, \mathcal{A})$ with VC-dimension ν, a random sample $T \subset P$ of size $O((1/\varepsilon^2)(\nu + \log(1/\delta)))$ is an ε-RC coreset of $(P, \mathcal{A})$ with probability at least $1 - \delta$.*

Yang et al. propose a similar result [155] as above, without proof.

2.6.1 RC Coresets in $\mathbb{R}^1$

Constructing ε-RC coresets when the family of ranges $\mathcal{I}_+$ represents one-sided, one-dimensional intervals is much easier than other cases. It relies heavily on the ordered structure of the canonical permutations, and thus discrepancy results do not need to decompose and then recompose the ranges.

Lemma 2.8. *A point $p_i \in P$ is in range $r \in \mathcal{I}_+$ with probability at least $\tau = t/k$ if and only if $p_{i,t} \in r \cap P_t$.*

Proof. By the canonical permutations, since for all $i \in [n]$, we require $p_{i,j} < p_{i,j+1}$, then if

$p_{i,t} \in r$, it follows that $p_{i,j} \in r$ for $j \leq t$. Similarly if $p_{i,t} \notin r$, then all $p_{i,j} \notin r$ for $j \geq t$. $\square$

Thus when each canonical permutation is represented upto an error ε by a coreset T, then each threshold τ is represented within ε. Hence, as with ε-RE coresets, we invoke the low-discrepancy coloring of Bohus [37] and Spencer *et al.* [146], and then iterate them (invoking Theorem 2.1) to achieve a small size ε-RC coreset.

Theorem 2.7. *For uncertain point set P and range space $(P_{cert}, \mathcal{I}_+)$ with ranges defined by one-sided intervals of the form $(-\infty, a]$. An ε-RC coreset of $(P, \mathcal{I}_+)$ can be constructed of size $O((\sqrt{k}/\varepsilon)\log(k/\varepsilon))$.*

Extending Lemma 2.8 from one-sided intervals of the form $[-\infty, a] \in \mathcal{I}_+$ to intervals of the form $[a, b] \in \mathcal{I}$ turns out to be nontrivial. It is *not* true that $G_{P,[a,b]}(\tau) = G_{P,[-\infty,b]}(\tau) - G_{P,[-\infty,a]}(\tau)$, hence the two queries cannot simply be subtracted. Also, while the set of points corresponding to the query $G_{P,[-\infty,a]}(\frac{t}{k})$ are a contiguous interval in the tth permutation we construct in Lemma 2.8, the same need not be true of points corresponding to $G_{P,[a,b]}(\frac{t}{k})$. This is a similar difficulty in spirit as noted by Kaplan et al. [110] in the problem of counting the number of points of distinct colors in a box where one cannot take a naive decomposition and add up the numbers returned by each subproblem.

We give now a construction to solve this two-sided problem for uncertain points in $\mathbb{R}^1$ inspired by that of Kaplan et al. [110], but we require specifying a fixed value of $t \in [k]$. Given an uncertain point $p_i \in P$ assume w.l.o.g that $p_{i,j} < p_{i,j+1}$. Also pretend there is a point $p_{i,k+1} = \eta$ where η is larger than any $b \in \mathbb{R}^1$ from a query range $[a, b]$ (essentially $\eta = \infty$). Given a range $[a, b]$, we consider the right-most set of t locations of p_i (here $\{p_{i,j-t}, \ldots, p_{i,j}\}$) that are in the range. This satisfies (i) $p_{i,j-t} \geq a$, (ii) $p_{i,j} \leq b$, and (iii) to ensure that it is the right-most such set, $p_{i,j+1} > b$.

To satisfy these three constraints we re-pose the problem in $\mathbb{R}^3$ to designate each contiguous set of t possible locations of p_i as a single point. So for $t < j \leq k$, we map $p_{i,j}$ to $\bar{p}^t_{i,j} = (p_{i,j-t}, p_{i,j}, p_{i,j+1})$. Correspondingly, a range $r = [a, b]$ is mapped to a range $\bar{r} = [a, \infty) \times (-\infty, b] \times (b, \infty)$; see Figure 2.2. Let $\bar{p}^t_i$ denote the set of all $\bar{p}^t_{i,j}$, and let $\bar{P}^t$ represent $\bigcup_i \bar{p}^t_i$.

Lemma 2.9. *p_i is in interval $r = [a, b]$ with threshold at least t/k if and only if $\bar{p}^t_i \cap \bar{r}^t \geq 1$.*

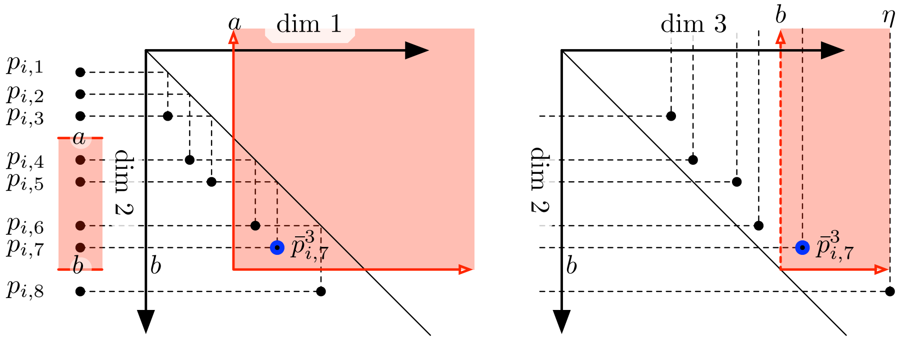


Figure 2.2: Uncertain point p_i queried by range $[a, b]$. Lifting shown to $\bar{p}_i^3$ along dimensions 1 and 2 (left) and along dimensions 2 and 3 (right).

Furthermore, no two points $p_{i,j}, p_{i,j'} \in p_i$ can map to points $\bar{p}^t_{i,j}, \bar{p}^t_{i,j'}$ such that both are in a range $\bar{r}^t$.

Proof. Since $p_{i,j} < p_{i,j+1}$, then if $p_{i,j-t} \geq a$ it implies all $p_{i,\ell} \geq a$ for $\ell \geq j - t$, and similarly, if $p_{i,j} \leq b$ then all $p_{i,\ell} \leq b$ for all $\ell \leq j$. Hence if $\bar{p}^t_{i,j}$ satisfies the first two-dimensional constraints of the range $\bar{r}^t$, it implies t points $p_{i,j-t} \ldots, p_{i,j}$ are in the range $[a, b]$. Satisfying the constraint of $\bar{r}^t$ in the third coordinate indicates that $p_{i,j+1} \notin [a, b]$. There can only be one point $p_{i,j}$ which satisfies the constraint of the last two coordinates that $p_{i,j} \leq b < p_{i,j+1}$. And for any range which contains at least t possible locations, there must be at least one such set (and only one) of t consecutive points which has this satisfying $p_{i,j}$. □

Corollary 2.2. *Any uncertain point set $P \in \mathbb{R}^1$ of size n and range $r = [a, b]$ has $G_{P,r}(\frac{t}{k}) = |\bar{P}^t \cap \bar{r}^t|/n$.*

This presents an alternative view of each uncertain point in $\mathbb{R}^1$ with k possible locations as an uncertain point in $\mathbb{R}^3$ with $k - t$ possible locations (since for now we only consider a threshold $\tau = t/k$). Where $\mathcal{I}$ represents the family of ranges defined by two-sided intervals, let $\bar{\mathcal{I}}$ be the corresponding family of ranges in $\mathbb{R}^3$ of the form $[a, \infty) \times (-\infty, b] \times (b, \infty)$ corresponding to an interval $[a, b] \in \mathcal{I}$. Under the assumption (valid under the lifting defined above) that each uncertain point can have at most one location fall in each range, we can now decompose the ranges and count the number of points that fall in each subrange and add them together. Using the techniques (described in detail in Section 2.5.4) of Bohus [37] and Spencer et al. [146] we can consider $\ell = (k - t)(1 + \lceil \log n \rceil)^2$ permutations of $\bar{P}^t_{\mathsf{cert}}$ such that each range $\bar{r} \in \bar{\mathcal{I}}$ can be written as the points in a disjoint union of intervals from these permutations. To extend low discrepancy to *each* of the k distinct values of threshold t, there are k such liftings and $h = k \cdot \ell = O(k^2 \log^2 n)$ such permutations we need to consider. We can construct a coloring $\chi : P \to \{-1, +1\}$ such that intervals on each permutation has discrepancy $O(\sqrt{h} \log n) = O(k \log^2 n)$. Recall that for any fixed threshold t we only need to consider the corresponding ℓ permutations. Hence the total discrepancy for any such range is at most the sum of discrepancy from all corresponding $\ell = O(k \log^2 n)$ permutations or $O(k^2 \log^4 n)$. Finally, this low-discrepancy coloring can be iterated (via Theorem 2.1) to achieve the following theorem.

Theorem 2.8. *Consider an uncertain point set P along with ranges $\mathcal{I}$ of two-sided intervals. We can construct an ε-RC coreset T for $(P, \mathcal{I})$ of size $O((k^2/\varepsilon) \log^4(k/\varepsilon))$.*

2.6.2 RC Coresets for Rectangles in $\mathbb{R}^d$

The approach for $\mathcal{I}$ can be further extended to $\mathcal{R}_d$, axis-aligned rectangles in $\mathbb{R}^d$. Again the key idea is to define a proxy point set $\bar{P}$ such that $|\bar{r} \cap \bar{P}|$ equals the number of uncertain points in r with at least threshold t. This requires a suitable lifting map and decomposition of space to prevent over or under counting; we employ techniques from Kaplan et al. [110].

First we transform queries on axis-aligned rectangles in $\mathbb{R}^d$ to the semibounded case in $\mathbb{R}^{2d}$. Denote the x_i-coordinate of a point q as $x_i(q)$, we double all the coordinates of each point $q = (x_1(q), ..., x_\ell(q), ..., x_d(q))$ to obtain point

$$\tilde{q} = (-x_1(q), x_1(q)..., -x_\ell(q), x_\ell(q), ..., -x_d(q), x_d(q))$$

in $\mathbb{R}^{2d}$. Now answering range counting query $\prod_{i=1}^d [a_i, b_i]$ is equivalent to solving the query

$$\prod_{i=1}^{d} [(-\infty, -a_i] \times (-\infty, b_i]]$$

on the lifted point set.

Based on this reduction we can focus on queries of *negative orthants* of the form $\prod_{i=1}^d (-\infty, a_i]$ and represent each orthant by its apex $a = (a_1, ..., a_d) \in \mathbb{R}^d$ as Q_a^-. Similarly, we can define Q_a^+ as *positive orthants* in the form $\prod_{i=1}^d [a_i, \infty) \subseteq \mathbb{R}^d$. For any point set $A \subset \mathbb{R}^d$ define $U(A) = \cup_{a \in A} Q_a^+$.

A *tight* orthant has *a* location of $p_i \in P$ incident to every bounding facet. Let $C_{i,t}$ be the set of all apexes representing tight negative orthants that contain exactly t locations of

p_i; see Figure 2.3(a). An important observation is that query orthant Q_a^- contains p_i with threshold at least t if and only if it contains at least one point from $C_{i,t}$.

Let $Q_{i,t}^+ = \cup_{c \in C_{i,t}} Q_c^+$ be the locus of all negative orthant query apexes that contain at least t locations of p_i; see Figure 2.3(b). Notice that $Q_{i,t}^+ = U(C_{i,t})$.

Lemma 2.10. *For any point set $p_i \subset \mathbb{R}^d$ of k points and some threshold $1 \leq t \leq k$, we can decompose $U(C_{i,t})$ into $f(k) = O(k^d)$ pairwise disjoint boxes, $B(C_{i,t})$.*

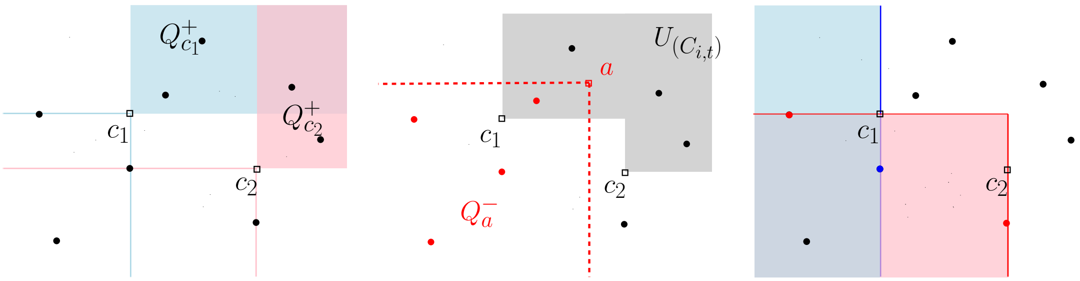
Proof. Let $M(A)$ be the set of maximal empty negative orthants for a point set A, such that any $m \in M(A)$ is also bounded in the positive direction along the 1st coordinate axis. Kaplan *et al.* [110] show (within Lemma 3.1) that $|M(A)| = |B(A)|$ and provide a specific construction of the boxes B. Thus we only need to bound $|M(C_{i,t})|$ to complete the proof; see $M(C_{i,t})$ in Figure 2.3(c). We note that each coordinate of each $c \in C_{i,t}$ must be the same as some $p_{i,j} \in p_i$. Thus for each coordinate, among all $c \in C_{i,t}$ there are at most k values. And each maximal empty tight orthant $m \in M(C_{i,t})$ is uniquely defined by the d coordinates along the axis direction each facet is orthogonal to. Thus $|M(C_{i,t})| \leq k^d$, completing the proof. □

Note that as we are working in a lifted space $\mathbb{R}^{2d}$, this corresponds to $U(C_{i,t})$ being decomposed into $f(k) = O(k^{2d})$ pairwise *disjoint* boxes in which d is the dimensionality of our original point set.

Lemma 2.11. *For negative orthant queries Q_a^- with apex a on uncertain point set P, a point $p_i \in P$ is in Q_a^- with probability at least t/k if a is in some box in $B(C_{i,t})$, and a will lie in at most one box from $B(C_{i,t})$.*

Proof. The query orthant Q_a^- contains point p_i with threshold at least t if and only if Q_a^- contains at least one point from $C_{i,t}$ and this happens only when $a \in U(C_{i,t})$. Since the union of constructed boxes in $B(C_{i,t})$ is equivalent to $U(C_{i,t})$ and they are disjoint, the result follows. □

Corollary 2.3. *The number of uncertain points from P in query range Q_a^- with probability at least t/k is exactly the number of boxes in $\cup_i^n B(C_{i,t})$ that contain a.*



(a) (b) (c)

Figure 2.3: Illustration of uncertain point $p_i \in \mathbb{R}^2$ with $k = 8$ and $t = 3$. (a): All tight negative orthants containing exactly $t = 3$ locations of p_i, their apexes are $C_{i,t} = \{c_1, c_2\}$. (b): $U(C_{i,t})$ is shaded and query Q^-_a. (c): $M(C_{i,t})$, the maximal negative orthants of $C_{i,t}$ that are also bounded in the x-direction.

Thus for a set of boxes representing P, we need to perform count stabbing queries with apex a and show a low-discrepancy coloring of boxes.

We do a second lifting by transforming each point $a \in \mathbb{R}^d$ to a semibounded box $\bar{a} = \prod_{i=1}^{d}((-\infty, a_i] \times [a_i, \infty))$ and each box $b \in \mathbb{R}^d$ of the form $\prod_i^d [x_i, y_i]$ to a point $\bar{b} = (x_1, y_1, ..., x_\ell, y_\ell, ..., x_d, y_d)$ in $\mathbb{R}^{2d}$. It is easy to verify that $a \in b$ if and only if $\bar{b} \in \bar{a}$.

Since this is our second doubling of dimension, we are now dealing with points in $\mathbb{R}^{4d}$. Lifting P to $\bar{P}$ in $\mathbb{R}^{4d}$ now presents an alternative view of each uncertain point $p_i \in P$ as an uncertain point $\bar{p}_i$ in $\mathbb{R}^{4d}$ with $g_k = O(k^{2d})$ possible locations with the query boxes represented as $\bar{\mathcal{R}}$ in $\mathbb{R}^{4d}$.

We now proceed similarly to the proof of Theorem 2.8. For a fixed threshold t, obtain $\ell = g_k \cdot (1 + \lceil \log n \rceil)^{4d-1}$ disjoint permutations of $\bar{P}^t_{\mathsf{cert}}$ such that each range $\bar{r} \in \bar{\mathcal{R}}$ can be written as the points in a disjoint union of intervals from these permutations. For the k distinct values of t, there are k such liftings and $h = O\left(k \cdot g_k \cdot \log^{4d-1} n\right)$ such permutations we need to consider, and we can construct a coloring $\chi : P \to \{-1, +1\}$ so that intervals on each permutation have discrepancy $O(\sqrt{h} \log n) = O(k^{d+\frac{1}{2}} \log^{\frac{4d+1}{2}} n)$.

Hence for any such range and specific threshold t, the total discrepancy is the sum of discrepancy from all corresponding $\ell = O\left(g_k \cdot \log^{4d-1} n\right)$ permutations, or $O\left(k^{3d+\frac{1}{2}} \log^{6d-\frac{1}{2}} n\right)$. By applying the iterated low-discrepancy coloring (Theorem 2.1), we achieve the following result.

Theorem 2.9. *Consider an uncertain point set P and range space $(P_{\mathsf{cert}}, \mathcal{R}_d)$ with ranges defined by axis-aligned rectangles in $\mathbb{R}^d$. Then an ε-RC coreset can be constructed of size*

$$O\left((k^{3d+\frac{1}{2}}/\varepsilon)\log^{6d-\frac{1}{2}}(k/\varepsilon)\right).$$

2.7 RQ Coresets

In this section, given an uncertain point set P and its ε-RE coreset T, we want to determine values ε' and α so T is an (ε', α)-RQ coreset. That is for any $r \in \mathcal{A}$ and threshold $\tau \in [0,1]$ there exists a $\gamma \in [\tau - \alpha, \tau + \alpha]$ such that

$$\left|\mathsf{Pr}_{Q \Subset P}\left[\frac{|Q \cap r|}{|Q|} \leq \tau\right] - \mathsf{Pr}_{S \Subset T}\left[\frac{|S \cap r|}{|S|} \leq \gamma\right]\right| \leq \varepsilon'.$$

At a high level, our tack will be to realize that both $|Q \cap r|$ and $|S \cap r|$ behave like Binomial random variables. By T being an ε-RE coreset of P, then after normalizing, its mean is at most ε-far from that of P. Furthermore, Binomial random variables tend to concentrate around their mean–and more so for those with more trials. This allows us to say $|S \cap r|/|S|$ is either α-close to the expected value of $|Q \cap r|/|Q|$ or is ε'-close to 0 or 1. Since $|Q \cap r|/|Q|$ has the same behavior, but with more concentration, we can bound their distance by the α and ε' bounds noted before. We now work out the details.

Theorem 2.10. *If T is an ε-RE coreset of P for $\varepsilon \in (0, 1/2)$, then T is an (ε', α)-RQ coreset for P for $\varepsilon', \alpha \in (0, 1/2)$ and satisfying $\alpha \geq \varepsilon + \sqrt{(1/2|T|)\ln(2/\varepsilon')}$.*

Proof. We start by examining a Chernoff-Hoeffding bound on a set of independent random variables X_i so that each $X_i \in [a_i, b_i]$ with $\Delta_i = b_i - a_i$. Then for some parameter $\beta \in (0, \sum_i \Delta_i/2)$

$$\mathsf{Pr}\left[\left|\sum_i X_i - \mathsf{E}\left[\sum_i X_i\right]\right| \geq \beta\right] \leq 2\exp\left(\frac{-2\beta^2}{\sum_i \Delta_i^2}\right).$$

Consider any $r \in \mathcal{A}$. We now identify each random variable $X_i = 1(q_i \in r)$ (that is, 1 if $q_i \in r$ and 0 otherwise) where q_i is the random instantiation of some $p_i \in T$. So $X_i \in \{0, 1\}$ and $\Delta_i = 1$. Thus by equating $|S \cap r| = \sum X_i$

$$\mathsf{Pr}_{S \Subset T}\left[||S \cap r| - \mathsf{E}\left[|S \cap r|\right]| \geq \beta|S|\right] \leq 2\exp\left(\frac{-2\beta^2|S|^2}{\sum_i \Delta_i^2}\right) = 2\exp(-2\beta^2|S|) \leq \varepsilon'.$$

Thus by solving for β (and equating $|S| = |T|$)

$$\mathsf{Pr}_{S \Subset T}\left[\left|\frac{|S \cap r|}{|S|} - \mathsf{E}\left[\frac{|S \cap r|}{|S|}\right]\right| \geq \sqrt{\frac{1}{2|T|}\ln(\frac{2}{\varepsilon'})}\right] \leq \varepsilon'.$$

Now by T being an ε-RE coreset of P then

$$\left|\mathbf{E}_{S \Subset T}\left[\frac{|S \cap r|}{|S|}\right] - \mathbf{E}_{Q \Subset P}\left[\frac{|Q \cap r|}{|Q|}\right]\right| \leq \varepsilon.$$

Combining these two we have

$$\mathbf{Pr}_{S \Subset T}\left[\left|\frac{|S \cap r|}{|S|} - \mathbf{E}_{Q \Subset P}\left[\frac{|Q \cap r|}{|Q|}\right]\right| \geq \alpha\right] \leq \varepsilon'$$

for $\alpha = \varepsilon + \sqrt{\frac{1}{2|T|}\ln(\frac{2}{\varepsilon'})}$.

Combining these statements, for any $x \leq M - \alpha \leq M - \alpha'$ we have $\varepsilon' > F_{T,r}(x) \geq 0$ and $\varepsilon' > F_{P,r}(x) \geq 0$ (and symmetrically for $x \geq M + \alpha \geq M + \alpha'$). It follows that $F_{T,r}$ is an (ε', α)-quantization of $F_{P,r}$.

Since this holds for any $r \in \mathcal{A}$, by T being an ε-RE coreset of P, it follows that T is also an (ε', α)-RQ coreset of P. □

We can now combine this result with specific results for ε-RE coresets to get size bounds for (ε, α)-RQ coresets. To achieve the below bounds we set $\varepsilon = \varepsilon'$.

Corollary 2.4. *For uncertain point set P with range space $(P_{\mathsf{cert}}, \mathcal{A})$, there exists a $(\varepsilon, \varepsilon + \sqrt{(1/2|T|)\ln(2/\varepsilon)})$-RQ coreset of $(P, \mathcal{A})$ of size $|T| =$*

- *$O((1/\varepsilon^2)(\nu + \log(k/\delta)))$ when $\mathcal{A}$ has VC-dimension ν, with probability $1 - \delta$ (Theorem 2.3),*
- *$O((\sqrt{k}/\varepsilon)\log(k/\varepsilon))$ when $\mathcal{A} = \mathcal{I}$ (Theorem 2.4), and*
- *$O\left((\sqrt{k}/\varepsilon)\log^{\frac{3d-1}{2}}(k/\varepsilon)\right)$ when $\mathcal{A} = \mathcal{R}_d$ (Theorem 2.5).*

Finally we discuss why the α term in the (ε', α)-RQ coreset T is needed. Recall from Section 2.5 that approximating the value of $\mathbf{E}_{Q \Subset P}\left[\frac{|Q \cap r|}{|Q|}\right]$ with $\mathbf{E}_{S \Subset T}\left[\frac{|S \cap r|}{|S|}\right]$ for all r corresponds to a low-discrepancy sample of P_{cert}. Discrepancy error immediately implies we will have at least the ε horizontal shift between the two distributions and their means, unless we could obtain a zero discrepancy sample of P_{cert}. Note this ε-horizontal error corresponds to the α term in an (ε', α)-RQ coreset. When P is very large, then due to the central limit theorem, $F_{P,r}$ will grow very sharply around $\mathbf{E}_{Q \Subset P}\left[\frac{|Q \cap r|}{|Q|}\right]$. In the worst case $F_{T,r}$ may be $\Omega(1)$ vertically away from $F_{P,r}$ on either side of $\mathbf{E}_{S \Subset T}\left[\frac{|S \cap r|}{|S|}\right]$, so no reasonable amount of ε' vertical tolerance will make up for this gap.

On the other hand, the ε' vertical component is necessary since for very small probability events (that is for a fixed range r and small threshold τ) on P, we may need a much smaller value of τ (smaller by $\Omega(1)$) to get the same probability on T, requiring a very large horizontal shift. But since it is a very small probability event, only a small vertical ε' shift is required.

The main result of this section then is showing that there exist pairs (ε', α) which are both small.

CHAPTER 3

SUBLINEAR ALGORITHMS FOR MAXCUT AND CORRELATION CLUSTERING

There is a long line of work on efficient approximation schemes for MaxCut and correlation clustering, which first we give an overview here. Then, we provide the problem statement and the technical details.

3.1 Prior Work

One direction by Arora et al. [22] is to get PTASs for the case of dense graphs. This subsequently improved by Fotakis et al. [81] to the polynomial density graphs we consider here, albeit in time $\exp(n^{1-\delta})$. Alon et al. [16] took a different approach, sampling the linear program formulation and analyzing the optimal solution on the subprogram, yielding similar bounds for dense graphs. In both approaches, full random access to the graph is needed. Other algorithms for the dense case are the ones from property testing (see [87]), and the work of Mathieu and Schudy [129], who showed that simple sampling combined with a greedy strategy improves the above bounds for MaxCut by a $\log(\frac{1}{\epsilon})$ factor.

In the "polynomial density" (average degree n^{δ}) case that we consider, Feige and Schectman [77] give a combinatorial approach to show that uniformly sampling vertices with probability roughly $1/\Delta$ preserves the value of the cut. Similar results were thoroughly explored in [33]. These works heavily rely on a "near-regularity" requirement on the graph (roughly speaking, that all degrees are within a small factor of each other), which can be used to obtain strong concentration bounds. Their algorithm makes use of a uniform sample of the vertices; this is in contrast to the biased sampling that we use in our more general setting (when degrees can vary widely). Indeed, the main challenge in our work is to deal with this issue, using biased sampling (which we analyze via the LP framework).

In the setting of streaming algorithms, there are several algorithms [10, 14, 85, 86, 114, 116] which produce cut or spectral sparsifiers with $O(\frac{n}{\epsilon^2})$ edges using $\tilde{O}(\frac{n}{\epsilon^2})$ bits space which preserves every cut within $(1+\epsilon)$-factor (and therefore also preserve the max cut). All these algorithms construct the sparsifier graph by sampling a subgraph of the original graph and reweighting the sampled edges appropriately. All of these approaches use space $\tilde{O}(n)$.

In [112], Kapralov et al. proved that any streaming algorithm that breaks the 2-approximation barrier requires $\tilde{O}(\sqrt{n})$ space, even if the edges of the streaming graph are presented in random order. They also show that for any $\epsilon > 0$, any streaming algorithm that obtains a $(1+\epsilon)$-approximation to the max cut value when edges arrive in adversarial order requires $n^{1-O(\epsilon)}$ space. Kapralov et al. [113] also proved that there exists $\epsilon^* > 0$ such that every randomized single-pass streaming algorithm that yields a $(1+\epsilon^*)$-approximation to the MAXCUT size must use $\Omega(n)$ space. For each lower bound, the result follows by constructing a (distribution over) sparse graphs (with $O(n)$ edges). Separately, Andoni et al. [18] gave a $\Omega(\frac{n}{\epsilon^2})$ lower bound for the bit complexity of a *sketch*. Since all of the upper bounds are based on sketching techniques, this result shows that these algorithms are essentially optimal (upto logarithmic factors).

Correlation clustering was first formulated by Bansal et al. [30] and has been studied extensively in both streaming and nonstreaming contexts. There are two variants of the problem – maximizing agreement and minimizing disagreement. While these are equivalent for exact optimization (their sum is a constant), they look very different under an approximation lens. Maximizing agreement typically admits constant factor approximations, but minimizing disagreement is much harder. In this paper, we focus on the maximizing-agreement variant of correlation clustering and in particular we focus on methods that seek $(1+\epsilon)$-approximations. Ailon and Karnin [15] presented an approximation scheme with sublinear query complexity (which also yields a semistreaming algorithm) for dense instance of correlation clustering. Giotis and Guruswami [83] described a sampling based algorithm combined with a greedy strategy which guarantees a solution within (ϵn^2) additive error for max-agreement (their work is similar to spirit to the sample-and-then-be-greedy technique used by Mathieu and Schudy [129]). Most recently, Ahn et al. [9] gave single-pass semistreaming algorithms for max-agreement. For input graphs with bounded weights, they provide an $(1+\epsilon)$-approximation streaming algorithm and for graphs with arbitrary weights, they present a $0.766(1-\epsilon)$-approximation streaming algorithm. Both algorithms require $(n\epsilon^{-2})$ space. The key idea in their approach was to adapt multiplicative-weight-update methods for solving the natural SDPs for correlation clustering in a streaming setting using linear sketching techniques. We mention in passing work by Chierichetti et al. [58] that while focusing on min-disagreement, is also a multipass semistreaming algorithm that yields a constant factor approximation on $[0, 1]$-weighted graphs.

3.2 Definitions and Preliminaries

Definition 3.1 (MaxCut). *Let $G = (V, E, w)$ be a graph with weights $w\colon E \to \mathbb{R}^+$. Let (A, B) be a partition of V and let $c(A, B)$ denote the sum of weights of edges going between A and B. Then*

$$\mathrm{MaxCut}(G) = \max_{(A,B) \text{ partition of } V} c(A, B)$$

In this paper, for ease of presentation, we will assume that the input graph is unweighted. However, our techniques apply generally as long as all weights are bounded by some constant.

Let $G = (V, E, c^+, c^-)$ be a graph with edge weights c^+_{ij} and c^-_{ij} where for every edge ij we have $c^+_{ij}, c^-_{ij} \geq 0$ and only one of them is nonzero. For every edge $ij \in E$, we define $\eta_{ij} = c^+_{ij} - c^-_{ij}$ and for each vertex, $d_i = \sum_{i \in \Gamma(j)} |\eta_{ij}|$. We will also assume that all the weights are bounded by an absolute constant in magnitude (for simplicity, we assume it is 1). We define the "average degree" Δ (used in the statements that follow) of a correlation clustering instance to be $(\sum_i d_i)/n$.

Definition 3.2 (MAX-AGREE correlation clustering). *Given $G = (V, E, c^+, c^-)$ as above, consider a partition of V into* clusters $C_1, C_2, \ldots, C_k$*, and let χ_{ij} be an indicator variable that is set to* 1 *if i, j are in the same cluster and* 0 *otherwise. The MAX-AGREE score of this clustering is given by*

$$\sum_{ij} c^+_{ij} \chi_{ij} + \sum_{ij} c^-_{ij} (1 - \chi_{ij})$$

and the goal is to find a partition maximizing *this score. The maximum value of this objective over all partitions of V will be denoted by $CC(G)$.*

Note that the objective value can be simplified to $\sum_{ij} c^-_{ij} + \eta_{ij}\chi_{ij} = C^- + \sum_{ij} \eta_{ij}\chi_{ij}$, where C^- denotes the sum $\sum_{ij} c^-_{ij}$.

We will frequently appeal to Bernstein's inequality for concentration of linear forms of random variables. For completeness, we state it here.

Theorem 3.3 (Bernstein's inequality [72])**.** *Let the random variables $X_1, \cdots, X_n$ be independent with $|X_i - E[X_i]| \leq b$ for each $i \in [n]$. Let $X = \sum_i X_i$ and let $\sigma^2 = \sum_i \sigma_i^2$ be the variance of X. Then, for any $t > 0$,*

$$\boldsymbol{Pr}[|X - \boldsymbol{E}[X]| > t] \leq \exp(-\frac{t^2}{2\sigma^2(1 + bt/3\sigma^2)})$$

A slightly more nonstandard concentration inequality we use is from Boucheron, Massart and Lugosi [38]. It can be viewed as an exponential version of the classic Efron-Stein lemma.

Theorem 3.4 ([38])**.** *Assume that $Y_1, \cdots, Y_n$ are random variables, and Y_1^n is the vector of these n random variables. Let $Z = f(Y_1, \cdots, Y_n)$, where $f : \chi_n \to R$ is a measurable function. Define $Z^{(i)} = f(Y_1, \cdots, Y_{i-1}, Y_i', Y_{i+1}, \cdots, Y_n)$, where $Y_1, \cdots, Y_n'$ denote the independent copies of $Y_1, \cdots, Y_n$. Then, for all $\theta > 0$ and $\lambda \in (0, \frac{1}{\theta})$,*

$$\log \boldsymbol{E}[e^{\lambda(Z - \boldsymbol{E}[Z])}] \leq \frac{\lambda\theta}{1 - \lambda\theta} \log \boldsymbol{E}[e^{\frac{\lambda L_+}{\theta}}],$$

where L_+ is the random variable defined as

$$L_+ = \boldsymbol{E}[\sum_{i=1}^{n} (Z - Z^{(i)})^2 \mathbf{1}_{Z > Z^{(i)}} | Y_1^n].$$

3.3 Technical Overview

The algorithm we develop is a simple biased sampling strategy followed by a reweighting of edges in the induced subgraph. Given a graph $G = (V, E)$, let d_i denote the degree of vertex $v_i \in V$.

Definition 3.5. *Let the* importance score *of each vertex v_i with degree d_i in graph G as $h_i = \min\{1, \frac{\max\{d_i, \epsilon\Delta\}}{\Delta^2 \alpha_\epsilon}\}$, where α_ϵ is an appropriately small constant (for* MaxCut*, it will suffice to choose it to be $\frac{\epsilon^4}{C \log n}$, and for correlation clustering, we choose it to be $\frac{\epsilon^6}{Ck^2 \log n}$, where C is an absolute constant).*

We start by introducing two sampling procedures that we will use in our algorithm.

- Procedure VERTEXSAMPLE. Let S' be the set formed by selecting each vertex v_i with probability p_i, where $p_i \geq h_i$. Then assign a weight $w_{ij} = \frac{1}{p_i p_j \Delta^2}$ to each edge ij in the induced subgraph H.[1]

  Intuitively, we sample a vertex with high enough probability, and then reweight the edges so that the edge counts are *unbiased*.
- Procedure EDGESAMPLE. Here we are given a weighted graph H, with total edge weight W. Now, sample each edge ij in H (independently) with probability $p_{ij} = \min(1, \frac{8|S'|w_{ij}}{\varepsilon^2 W})$, obtaining the graph H'. Now, assign a weight w_{ij}/p_{ij} to every edge in H'.

  Intuitively, we sample roughly $|S'|/\varepsilon^2$ edges proportional to their weights, while keeping the expected sum of edge weights the same. The main results of the paper can now be stated as follows.

Theorem 3.6 (Core set)**.** *Let G be a graph with average degree Δ. Suppose we apply* VERTEXSAMPLE *with probabilities $p_i \in [h_i, 2h_i]$ to obtain a weighted graph H. Then H has $\widetilde{O}(\frac{n}{\Delta})$ vertices and the quantities* $\mathrm{MaxCut}(H)$ *and $CC(H)$ are within a $(1+\epsilon)$ factor of the corresponding quantities* $\mathrm{MaxCut}(G)$ *and $CC(G)$, w.p. at least $1 - \frac{1}{n^2}$.*

Theorem 3.7 (Sparse core set)**.** *Given an input graph G with n vertices and average degree $\Delta = n^{\delta}$. If we apply* VERTEXSAMPLE *and then* EDGESAMPLE *to G, then the resulting graph H' is a sublinear sized ϵ-coreset for* MaxCut *and CC, having size $\widetilde{O}(\frac{n}{\Delta}) = \widetilde{O}(n^{1-\delta})$.*

3.3.1 Overview of the Proofs

Let us now give a rough outline of the proof. For this outline, we will restrict to the case of MAXCUT, as it illustrates almost all the ideas. Technically, the key step is proving Theorem 3.6. The sparse coreset results then follow via standard edge-sampling ideas. Let us thus consider the vertex sampling step, which outputs the graph H. We wish to reason about $\mathrm{MaxCut}(H)$.

The easy part is to show that the $\mathrm{MaxCut}(H)$ (if weighted correctly) is unlikely to be too small compared to $\mathrm{MaxCut}(G)$.

[1] In correlation clustering, we have edge weights to start with, so the weight in H will be $w_{ij} \cdot c_{ij}^{+}$ (or c_{ij}^{-}).

Lemma 3.1. *Let H be the graph produced by* VERTEXSAMPLE. *Then w.p. at least* $1 - \frac{1}{n^4}$,

$$\text{MaxCut}(H) \geq \text{MaxCut}(G) - \epsilon n \Delta$$

Intuitively, this is because we can look at the projection of the maximum cut in G onto the subgraph H. A simple concentration bound can be employed to show that the value of this cut is close to its expectation (see [77]). The main challenge is to argue that the cut in H cannot be too *large*. The key to showing this is the following "double sampling" argument, which has been used in works on property testing [87] and by Feige and Schechtman [77].

3.3.2 Two Views of Sampling

Consider the following two strategies for sampling a *pair* of subsets (S, S') of a universe $[n]$ (here, $q_v \leq p_v$ for all v):

- Strategy A: choose $S' \subseteq [n]$, by including every v w.p. p_v, independently; then for $v \in S'$, include them in S w.p. q_v / p_v, independently.
- Strategy B: pick $S \subseteq [n]$, by including every v w.p. q_v; then iterate over $[n]$ once again, placing $v \in S'$ with a probability equal to 1 if $v \in S$, and p^*_v if $v \notin S$.

Lemma 3.2. *Suppose $p^*_v = p_v(1 - \frac{q_v}{p_v})/(1 - p_v)$. Then the distribution on pairs (S, S') obtained by strategies A and B are identical.*

Proof. Let us examine strategy A. It is clear that the distribution over S is precisely the same as the one obtained by strategy B, since in both the cases, every v is included in S independently of the other v, with probability precisely q_v. Now, to understand the joint distribution (S, S'), we need to consider the conditional distribution of S' given S. Firstly, note that in both strategies, $S \subseteq S'$, i.e., $\mathbf{Pr}[v \in S' | v \in S] = 1$. Next, we can write $\mathbf{Pr}_{\text{strategy A}}[v \in S' | v \notin S]$ as

$$\frac{\mathbf{Pr}_{\text{strategy A}}[v \in S' \wedge v \notin S]}{\mathbf{Pr}_{\text{strategy A}}[v \notin S]} = \frac{p_v(1 - \frac{q_v}{p_v})}{1 - p_v}.$$

Noting that $\mathbf{Pr}_{\text{strategy}B}[v \in S' | v \notin S] = p^*_v$ (by definition) concludes the proof. □

In our proof of Theorem 3.6, we will set q to be the uniform distribution $q_v = \frac{16 \log n}{\varepsilon^2 \Delta}$. The proof now proceeds as follows. Let H denote the weighted graph on S' obtained by applying VERTEXSAMPLE to G. The goal is to show that $\text{MaxCut}(G) \approx \Delta^2 \text{MaxCut}(H)$, w.h.p.

3.3.3 Estimation with Linear Programs

The main idea is to establish the above approximate equality indirectly, by using a sampling based estimator for MaxCut, introduced by [22]. At a very high level, a set of "seed vertices" is used to obtain approximations to appropriate linear functions of the variables, which then allow us to reduce solving a quadratic program to solving a linear program. More concretely, the estimator takes as input the graph G and a set of sampling probabilities γ, and then uses γ to sample a small seed set of vertices S that can be used to encode a linear program. Enumerating over all assignments to S then yields the desired estimator $\text{Est}(G)$. We describe this estimator in detail in Section 3.4; it is a weighted version of the formulation of Arora et al. [22], introduced for approximating dense constraint satisfaction problems (like MaxCut).

Suppose we now apply VERTEXSAMPLE to G, yielding a graph H with (sampled) vertex set S'. We can now apply the estimation procedure $\text{Est}(H)$ with the sampled seed set S. Note that the sequence of first sampling S' and then S is the strategy A above. Corollary 3.2 then shows that $\text{MaxCut}(H) \approx \text{Est}(H)$.

We can also apply estimation to the original graph G. This can be thought of as strategy A, where S' is ignored. Again, we invoke Corollary 3.1 to show that $\text{MaxCut}(G) \approx \text{Est}(G)$.

But how do we compare the two estimation procedures? The difficulty here is that when computing $\text{Est}(H)$ we sample the subset S' and then sample the seed set S from S'. However, when computing $\text{Est}(G)$, the seed set S is sampled from V. Ideally, we would then just sample G to yield the subset S', but we have to make sure that S' contains S. Otherwise our estimation procedure is meaningless.

In effect, we need strategy B to compare the two estimators. Lemma 3.2 guarantees that strategy B is equivalent to strategy A, and we can choose to interpret our sampling strategy via strategy B as long as we adjust the sampling probabilities as prescribed by the Lemma.

Specifically, we assume that the seed set S is fixed and is the same for H and G. Then all that remains is to sample $S' \supset S$ (via strategy B) and show that

$$\text{Est}(G) \approx \Delta^2 \text{Est}(H).$$

This final step is done in Theorem 3.11 (in Section 3.5) which is the technical core of our result. It is essentially a theorem that relates the value of an LP to that of a random *induced* sub-LP (and is thus a concentration bound, at its core).

3.4 Estimation via Linear Programming

We now define a generalized estimation procedure that makes use of a linear programming formulation of the problem to yield an accurate estimate of the optimal solution. Our approach is based on the procedure outlined by Arora et al. [22]. The main difference here is that we have weighted graphs, and sampling is done according to an arbitrary distribution γ, as opposed to uniform.

Let $H = (V, E, w)$ be a weighted, undirected graph with edge weights w_{ij}, and let $\gamma : V \to [0, 1]$ denote sampling probabilities. Construct $S \subset V$ by including $v_i \in V$ with probability γ_i. Fix a partition $(A, S \setminus A)$ of S. Define the linear program $LP_{A,S\setminus A}(V)$ as

$$\begin{aligned} \text{maximize} \quad & \sum_i x_i(d_i - \rho_i) - s_i - t_i \\ \text{subject to} \quad & \rho_i - t_i \le \sum_{j\in\Gamma(i)} w_{ij}x_j \le \rho_i + s_i \\ 0 \le x_i \le 1, \quad & \quad s_i, t_i \ge 0. \end{aligned}$$

where

$$\rho_i = \sum_{j\in\Gamma(i)\cap A} \frac{w_{ij}}{\gamma_j} \text{ and } d_i = \sum_{j\in\Gamma(i)} w_{ij}.$$

The intuition here is that the set S represents a set of *witnesses* for which we fix a cut, and we then determine the assignment of the remaining vertices by looking only at a vertex's neighborhood in S. Note that the best choice of s_i, t_i for each i are such that $s_i + t_i = |\rho_i - \sum_{j\in\Gamma(i)} w_{ij}x_j|$.

Now define AKK-Est(H, γ) to be the output of the following randomized algorithm: sample S as above; *for each partition* $(A, S \setminus A)$ of S, compute $LP_{A,S\setminus A}(V)$, and return the largest value.

Theorem 3.8. *Let H be a weighted graph on n vertices, with edge weights w_{ij} that add up to W. Suppose the sampling probabilities γ_i satisfy the condition*

$$w_{ij} \le \frac{W\varepsilon^2}{8\log n}\frac{\gamma_i\gamma_j}{\sum_u \gamma_u} \quad \textit{for all } i, j. \tag{3.1}$$

Then, we have AKK-Est$(H, \gamma) \in$ MaxCut$(H) \pm \varepsilon W$*, with probability at least* $1 - 1/n^2$ *(where the probability is over the random choice of S).*

Proof. The proof consists of two claims as follow, that imply the necessary upper and lower bound.

Claim 3.9. *The estimate is not too small.*

Proof. We can show that with high probability over the choice of S, there exists a cut $(A, S \setminus A)$ of S such that $LP_{A,S\setminus A}(H) \geq \text{MaxCut}(H) - \varepsilon W$. The result follows by an application of Bernstein's inequality. Let $(A_H, V \setminus A_H)$ be the max cut in the full graph H. Now consider a sample S, and let $(A, S \setminus A)$ be its projection onto S. For any vertex i, recall that $\rho_i = \sum_{j\in\Gamma(i)\cap A} \frac{w_{ij}}{\gamma_j} = \sum_{j\in\Gamma(i)\cap A_H} Y_j \frac{w_{ij}}{\gamma_j}$, where Y_j is the indicator for $j \in S$. Thus

$$\mathbf{E}[\rho_i] = \sum_{j\in\Gamma(i)\cap A_H} \gamma_j \frac{w_{ij}}{\gamma_j} = \sum_{j\in\Gamma(i)\cap A_H} w_{ij}.$$

We will use Bernstein's inequality to bound the deviation in ρ_i from its mean. To this end, note that the variance can be bounded as

$$\mathbf{Var}[\rho_i] = \sum_{j\in\Gamma(i)\cap A_H} \gamma_j(1-\gamma_j)\frac{w_{ij}^2}{\gamma_j^2} \leq \sum_{j\in\Gamma(i)} \frac{(1-\gamma_j)w_{ij}^2}{\gamma_j}.$$

In what follows, let us write $d_i = \sum_{j\in\Gamma(i)} w_{ij}$ and $f_i = \frac{W\gamma_i}{\sum_u \gamma_u}$. Then, for every j, our assumption on the w_{ij} implies that $\frac{w_{ij}}{\gamma_j} \leq \frac{\varepsilon^2}{8\log n} f_i$. Thus, summing over j, we can bound the variance by $\frac{\varepsilon^2 d_i f_i}{8\log n}$. Now, using Bernstein's inequality (Theorem 3.3),

$$\mathbf{Pr}[|\rho_i - \mathbf{E}[\rho_i]| > t] \leq \exp\left(-\frac{t^2}{\frac{\varepsilon^2 d_i f_i}{4\log n} + \frac{2t}{3}\frac{\varepsilon^2 f_i}{8\log n}}\right). \tag{3.2}$$

Setting $t = \epsilon(d_i + f_i)$, and simplifying, we have that the probability above is $< \exp(-4\log n) = \frac{1}{n^4}$. Thus, we can take a union bound over all $i \in V$, and conclude that w.p. $\geq 1 - \frac{1}{n^3}$,

$$\left|\rho_i - \sum_{j\in\Gamma(i)\cap A_H} w_{ij}\right| \leq \varepsilon(d_i + f_i) \quad \text{for all } i \in V. \tag{3.3}$$

For any S that satisfies the above, consider the solution x that sets $x_i = 1$ for $i \in A_H$ and 0 otherwise. We can choose $s_i + t_i = |\rho_i - \sum_{j\in\Gamma(i)} w_{ij}x_j| \leq \varepsilon(d_i + f_i)$, by the above reasoning (eq. (3.3)). Thus the LP objective can be lower bounded as

$$\sum_i x_i(d_i - \rho_i) - \varepsilon(d_i + f_i) \geq \sum_i x_i(d_i - \sum_{j\in\Gamma(i)} w_{ij}x_j) - 2\varepsilon(d_i + f_i).$$

This is precisely $\text{MaxCut}(G) - 2\varepsilon \sum_i (d_i + f_i) \geq \text{MaxCut}(G) - 4\varepsilon W$. This completes the proof of the claim. □

Claim 3.10. *The estimate is not much larger than an optimal cut.*

Proof. Consider any feasible solution to the LP above (for some values ρ_i, s_i, t_i). There exists a cut in H of value at least the LP objective. The key insight here is that the linear program is derived from a natural quadratic objective function. However, this objective is a *linear* function of any x_i if all other x_j are fixed. This means that we can deterministically round any fractional solution one variable at a time so that the resulting solution lies in $0, 1^n$ without decreasing the objective value. Here, we have a feasible solution x to the LP, of objective value $\sum_i x_i(d_i - \rho_i) - \sum_i |\rho_i - \sum_{j\in\Gamma(i)} w_{ij}x_j|$, and we wish to move to a cut of at least this value. To this end, define the quadratic form

$$Q(x) := \sum_i x_i\big(d_i - \sum_{j\in\Gamma(i)} w_{ij}x_j\big).$$

The first observation is that for any $x \in [0,1]^n$, and any real numbers ρ_i, we have

$$Q(x) \geq \sum_i x_i(d_i - \rho_i) - \sum_i |\rho_i - \sum_{j\in\Gamma(i)} w_{ij}x_j|.$$

This is true simply because $Q(x) = \sum_i x_i\big(d_i - \rho_i\big) + x_i\big(\rho_i - \sum_{j\in\Gamma(i)} w_{ij}x_j\big)$, and the fact that the second term is at least $-|\rho_i - \sum_{j\in\Gamma(i)} w_{ij}x_j|$, as $x_i \in [0,1]$.

Next, note that the maximum of the form $Q(x)$ over $[0,1]^n$ has to occur at a boundary point, since for any fixing of variables other than a given x_i, the form reduces to a linear function of x_i, which attains maximum at one of the boundaries. Using this observation repeatedly lets us conclude that there is a $y \in \{0,1\}^n$ such that $Q(y) \geq Q(x)$. Since any such y corresponds to a cut, and $Q(y)$ corresponds to the cut value, the claim follows.[2] □

Combining the two claims gives us the desired upper and lower bounds for AKK-Est(H, γ) in terms of the optimal solution. □

We can now state two corollaries to Theorem 3.8 that imply good estimates for the MaxCut.

Corollary 3.1. *Let H in the framework be the original graph G, and let $\gamma_i = \frac{16\log n}{\varepsilon^2\Delta}$ for all i. Then the condition $w_{ij} \leq \frac{\varepsilon^2}{8\log n}\cdot\frac{W\gamma_i\gamma_j}{\sum_u \gamma_u}$ holds for all i, j, and therefore* AKK-Est$(G,\gamma) \in$ MaxCut$(G) \pm \epsilon W$*, w.p.* $\geq 1 - \frac{1}{n^2}$.

[2] We note that the proof in [22] used randomized rounding to conclude this claim, but this argument is simpler; also, later papers such as [81] used such arguments for derandomization.

The proof is immediate (with a slack of 2), as $w_{ij} = 1$, $W = n\Delta$, and all γ_u are equal.

Corollary 3.2. *Let H be the weighted sampled graph obtained from* VERTEXSAMPLE, *and let $\gamma_i = \frac{16 \log n}{\epsilon^2 \Delta} \frac{1}{p_i}$. Then the condition* (3.1) *holds w.p. $\geq 1 - n^{-3}$, and therefore* $\text{AKK-Est}(H, \gamma) \in \text{MaxCut}(H) \pm \epsilon W$ *w.p. $\geq 1 - n^{-2}$.*

Proof. In this case, we have $w_{ij} = \frac{1}{p_i p_j \Delta^2}$. Thus, simplifying the condition, we need to show that

$$\frac{1}{p_i p_j \Delta^2} \leq \frac{2W}{p_i p_j \Delta} \frac{1}{\sum_{u \in H} \frac{1}{p_u}}.$$

Now, for H sampled via probabilities p_i, we have (in expectation) $W = \frac{n}{\Delta}$, and $\sum_{u \in H} \frac{1}{p_u} = n$. A straightforward application of Bernstein's inequality yields that $W \geq \frac{n}{2\Delta}$ and $\sum_{u \in H} \frac{1}{p_u} \leq 2n$, w.p. at least $1 - n^{-3}$. This completes the proof. □

3.5 Random Induced Linear Programs

We will now show that the AKK-Est on H has approximately the same value as the estimate on G (with appropriate γ values). First, note that the estimate for $\text{MaxCut}(G)$ is $\max_{A \subseteq S} LP^{\gamma}_{A, S \setminus A}(G)$, where $\gamma_i = q_i$. To write the LP, we need the constants ρ_i, defined by the partition $(A, S \setminus A)$ as $\rho_i := \sum_{j \in \Gamma(i) \cap A} \frac{1}{q_j}$.

The LP is now as follows.

$$\begin{aligned} \text{maximize} \quad & \sum_{i \in G} [x_i(d_i - \rho_i) - (s_i + t_i)] \\ \text{subject to} \quad & \sum_{j \in \Gamma(i)} x_j \leq \rho_i + s_i, \quad \forall i \in [n] && (3.4) \\ & - \sum_{j \in \Gamma(i)} x_j \leq -\rho_i + t_i, \quad \forall i \in [n] && (3.5) \\ & 0 \leq x_i \leq 1 \quad \forall i \in [n] \end{aligned}$$

For the graph H, the estimation procedure uses an identical program, but the sampling probabilities are now $\alpha_i := q_i / p_i$, and the estimates ρ, which we now denote by $\widetilde{\rho}_i$, are defined by

$$\widetilde{\rho}_i := \sum_{j \in \Gamma(i) \cap A} \frac{p_j w_{ij}}{q_j}.$$

Note that by the way we defined w_{ij}, $\widetilde{\rho}_i = \frac{\rho_i}{p_i \Delta^2}$. The degrees are now $\widetilde{d}_i := \sum_{j \in \Gamma(i) \cap S'} w_{ij} = \sum_{j \in \Gamma(i) \cap S'} \frac{1}{p_i p_j \Delta^2}$. The LP is thus

$$\begin{aligned}
\text{maximize} \quad & \sum_{i \in S'} [x_i(\widetilde{d_i} - \widetilde{\rho}_i) - (\tilde{s}_i + \tilde{t}_i)] \\
\text{subject to} \quad & \sum_{j \in \Gamma(i) \cap S'} w_{ij} x_j \leq \widetilde{\rho}_i + \tilde{s}_i, \quad \forall i \in S' \\
& - \sum_{j \in \Gamma(i) \cap S'} w_{ij} x_j \leq -\widetilde{\rho}_i + \tilde{t}_i, \quad \forall i \in S' \\
& 0 \leq x_i \leq 1, \quad \tilde{s}_i, \tilde{t}_i \geq 0 \quad \forall i \in S'
\end{aligned}$$

Our aim in this section is to show the following:

Theorem 3.11. *Let G be an input graph, and let (S, S') be sampled as described in Section 3.3.1. Then, with probability $\geq 1 - \frac{1}{n^2}$, we have*

$$\max_{A \subseteq S} LP^{\gamma}_{A, S \setminus A}(G) \geq \Delta^2 \cdot \max_{A \subseteq S} LP^{\alpha}_{A, S \setminus A}(H) - \varepsilon n \Delta.$$

Proof outline. To prove the theorem, the idea is to take the "strategy B" viewpoint of sampling (S, S'), i.e., fix S, and sample S' using the probabilities p^*. Then, the theorem amounts to showing that the optimum value of a random 'induced linear program' is not much larger than the optimum value of the full program. This is shown by considering the duals of the programs, and constructing a feasible solution to the induced dual whose cost is not much larger than the dual of the full program, w.h.p. This implies the result, by linear programming duality.

Let us thus start by understanding the dual LP of $LP^{\gamma}_{A, S \setminus A}(G)$ given A (the variable z is the difference of the dual variables corresponding to (3.4) and (3.5) in the primal):

$$\begin{aligned}
\text{minimize} \quad & \sum_{i \in G} u_i + \rho_i z_i \\
\text{subject to} \quad & u_i + \sum_{j \in \Gamma(i)} z_j \geq d_i - \rho_i \\
& u_i \geq 0, \quad -1 \leq z_i \leq 1 \quad \forall i \in [n]
\end{aligned} \tag{3.6}$$

We note that for any given z, the optimal choice of u_i is $\max\{0, d_i - \rho_i - \sum_{j \in \Gamma(i)} z_j\}$; thus we can think of the dual solution as being the vector z. The optimal u_i may thus be bounded by $2d_i$, a fact that we will use later. Next, we write down the dual of the induced program, $LP^{\alpha}_{A, S \setminus A}(H)$.

$$\begin{aligned} \text{minimize} \quad & \sum_{i \in S'} [\tilde{u}_i + \widetilde{\rho}_i \tilde{z}_i] \\ \text{subject to} \quad & \tilde{u}_i + \sum_{j \in \Gamma(i) \cap S'} w_{ij} \tilde{z}_j \geq \widetilde{d}_i - \widetilde{\rho}_i \quad \forall i \in S' \\ & \tilde{u}_i \geq 0, \quad -1 \leq \tilde{z}_i \leq 1 \quad \forall i \in S'. \end{aligned} \tag{3.7}$$

Following the outline above, we will construct a feasible solution to LP (3.7), whose cost is close to the optimal dual solution to LP (3.6). The construction we consider is very simple: if z is the optimal dual solution to (3.6), we set $\tilde{z}_i = z_i$ for $i \in S'$ as the candidate solution to (3.7). This is clearly feasible, and thus we only need to compare the solution costs. The dual objective values are as follows

$$\mathsf{Dual}_G = \sum_{i \in V} \rho_i z_i + \max\{0,\, d_i - \rho_i - \sum_{j \in \Gamma(i)} z_j\} \tag{3.8}$$

$$\mathsf{Dual}_H \leq \sum_{i \in S'} \widetilde{\rho}_i z_i + \max\{0,\, \widetilde{d}_i - \widetilde{\rho}_i - \sum_{j \in \Gamma(i) \cap S'} w_{ij} z_j\} \tag{3.9}$$

Note that there is a $\leq$ in (3.9), as $\tilde{z}_i = z_i$ is simply one feasible solution to the dual (which is a minimization program). Next, our goal is to prove that w.p. at least $1 - \frac{1}{n^2}$,

$$\max_{A \subseteq S} \mathsf{Dual}_H \leq \max_{A \subseteq S} \mathsf{Dual}_G + \frac{\varepsilon n}{\Delta}.$$

Note that here, the probability is over the choice of S' *given* S (as we are taking view-B of the sampling). The first step in proving the above is to move to a slight variant of the quantity Dual_H, which is motivated by the fact that $\mathbf{Pr}[Y_i = 1]$ is not quite p_i, but p_i^* (as we have conditioned on S). Let us define $\widetilde{\rho}_i^* := \frac{\rho_i}{p_i^* \Delta^2}$ (recall that $\widetilde{\rho}_i$ is $\frac{\rho_i}{p_i \Delta^2}$), and $w_{ij}^* := \frac{1}{p_i^* p_j^* \Delta^2}$. So also, let $d_i^* := \sum_{j \in \Gamma(i)} Y_j w_{ij}^*$. Then, define

$$\mathsf{Dual}_H^* := \sum_{i \in S'} \widetilde{\rho}_i^* z_i + \max\{0,\, \widetilde{d}_i^* - \widetilde{\rho}_i^* - \sum_{j \in \Gamma(i) \cap S'} w_{ij}^* z_j\}. \tag{3.10}$$

A straightforward lemma is the following, which bounds the difference between the "corrected" dual we used to analyze, and the value we need for the main theorem.

Lemma 3.3. *Let (S, S') be sampled as described in Section 3.3.1. Then w.p. at least $1 - \frac{1}{n^4}$, we have that for all $z \in [-1, 1]^n$ and for all partitions $(A, S \setminus A)$ of S,*[3]

$$|\mathit{Dual}_H - \mathit{Dual}_H^*| \leq \frac{\varepsilon n}{2\Delta}.$$

[3] Note that the partition defines the ρ_i.

Proof. To prove the lemma, it suffices to prove that w.p. $\geq 1 - \frac{1}{n^4}$,

$$\sum_i Y_i |\widetilde{\rho}_i - \widetilde{\rho}_i^*| + \sum_i Y_i \sum_{j \in \Gamma(i)} Y_j |w_{ij} - w_{ij}^*| \leq \frac{\varepsilon n}{2\Delta}. \tag{3.11}$$

This is simply by using the fact that z_i are always in $[-1, 1]$. Before showing this, we introduce some notation and make some simple observations. First, denote by Y the indicator vector for S' and by X the indicator for S.

Observation 3.12. *With probability $\geq 1 - \frac{1}{n^4}$ over the choice of (S, S'), we have:*

1. *For all $i \in V$, $\sum_{j \in \Gamma(i)} \frac{X_j}{q_j} \leq 2(d_i + \varepsilon\Delta)$.*
2. *For all $i \in V$, $\sum_{j \in \Gamma(i), j \notin S} \frac{Y_j}{p_j^*} \leq 2(d_i + \varepsilon\Delta)$.*
3. *$\sum_i \frac{X_i(d_i + \varepsilon\Delta)}{p_i} \leq 2\varepsilon^2 n\Delta$.*
4. *$\sum_{i \notin S} \frac{Y_i(d_i + \varepsilon\Delta)}{p_i^*} \leq 2n\Delta$.*

All the inequalities are simple consequences of Bernstein's inequality (and our choice of parameters p_i, p_i^*, q_i), and we thus skip the proofs. Next, note that as an immediate consequence of part-1, we have

$$\rho_i \leq 2(d_i + \varepsilon\Delta), \qquad \text{for all partitions } (A, S \setminus A) \text{ of } S. \tag{3.12}$$

Also, note that from the definitions of the quantities (and the fact $q_i/p_i \leq \varepsilon^2$), we have

$$\forall i \notin S, \quad \left| \frac{1}{p_i} - \frac{1}{p_i^*} \right| \leq \frac{\varepsilon^2}{p_i^*} \tag{3.13}$$

Now, we are ready to show (3.11). The first term can be bounded as follows:

$$\sum_i Y_i |\widetilde{\rho}_i - \widetilde{\rho}_i^*| = \sum_i \frac{Y_i \rho_i}{\Delta^2} \left| \frac{1}{p_i} - \frac{1}{p_i^*} \right| = \sum_{i \in S} \frac{\rho_i}{\Delta^2} \left| \frac{1}{p_i} - 1 \right| + \sum_{i \notin S} \frac{Y_i \rho_i}{\Delta^2} \left| \frac{1}{p_i} - \frac{1}{p_i^*} \right|. \tag{3.14}$$

Using (3.12) and part-3 of the observation, the first term can be bounded by $O(\varepsilon^2 n/\Delta)$. For the second term, using (3.13) together with part-4 of the observation gives a bound of $O(\varepsilon^2 n/\Delta)$. Thus the RHS above is at most $\frac{\varepsilon n}{16\Delta}$, as we may assume ε is small enough.

Now, consider the second term in (3.11). When $i \notin S$ and $j \notin S$, we have $|w_{ij} - w_{ij}^*|$ being "small". We can easily bound by $2\varepsilon^2 w_{ij}^*$, using

$$\left| \frac{1}{p_i p_j} - \frac{1}{p_i^* p_j^*} \right| \leq \left| \frac{1}{p_i p_j} - \frac{1}{p_i^* p_j} \right| + \left| \frac{1}{p_i^* p_j} - \frac{1}{p_i^* p_j^*} \right| \leq \frac{\varepsilon^2}{p_i^* p_j} + \frac{\varepsilon^2}{p_i^* p_j^*} \leq \frac{2\varepsilon^2}{p_i^* p_j^*}.$$

In the last steps, we used (3.13) and the fact that $p_i^* \leq p_i$ for $i \notin S$.

For $i \in S$ or $j \in S$, we can simply bound $|w_{ij} - w^*_{ij}|$ by $2w_{ij}$. Thus we can bound the second term in (3.11) as

$$4\sum_{i \in S} \frac{1}{p_i \Delta^2} \sum_{j \in \Gamma(i)} \frac{Y_j}{p_j} + \sum_{i \notin S} \frac{Y_i}{p^*_i \Delta^2} \sum_{j \in \Gamma(i) \setminus S} \frac{\varepsilon^2 Y_j}{p_j}.$$

The second term has only a sum over j not in S – this is why have an extra 2 factor for the first term. Now, consider the first term. The inner summation can be written as $\sum_{j \in \Gamma(i) \cap S} \frac{1}{p_j} + \sum_{j \in \Gamma(i) \setminus S} \frac{Y_j}{p_j}$. Using parts 1 and 2 of the observation, together with $p_j \geq q_j / \alpha_\varepsilon$, and $p_j \geq p^*_j$ for $j \notin S$, we have $\sum_{j \in \Gamma(i)} \frac{Y_j}{p_j} \leq 4\alpha_\varepsilon(d_i + \varepsilon\Delta)$. Then, using part-3 gives the desired bound on the first term.

Let us thus consider the second term. Again using part 2 along with $p_j \geq p^*_j$ for $j \notin S$, we can bound the inner sum by $2\varepsilon^2(d_i + \varepsilon\Delta)$. Then we can appeal to part-4 of the observation to obtain the final claim.

This ends up bounding the second term of (3.11), thus completing the proof of the lemma. □

Thus our goal in the rest of the section is to show:

Lemma 3.4. *Let S satisfy the conditions (a) $|S| \leq \frac{20 n \log n}{\varepsilon^2 \Delta}$, and (b) for all $i \in V$, $\sum_{j \in \Gamma(i) \cap S} \frac{1}{q_j} \leq 2(d_i + \varepsilon\Delta)$. Then, w.p. $\geq 1 - \frac{1}{n^4}$ over the choice of S' given S, we have*

$$\max_{A \subseteq S} \mathsf{Dual}^*_H \leq \max_{A \subseteq S} \mathsf{Dual}_G + \frac{\varepsilon n}{2\Delta}.$$

The condition (b) on S is a technical one that lets us bound ρ_i in the proofs. Let us see why the lemma implies Theorem 3.11.

Proof of Theorem 3.11. The conditions we assumed on S in Lemma 3.4 hold w.p. at least $1 - \frac{1}{n^4}$ (via a simple application of Bernstein's inequality). Thus the conclusion of the lemma holds w.p. at least $1 - \frac{2}{n^4}$. Combining this with Lemma 3.3, we have that $\max_A \mathsf{Dual}_H \leq \max_A \mathsf{Dual}_G + \frac{\varepsilon n}{\Delta}$ w.p. at least $1 - \frac{3}{n^4}$. The theorem then follows via LP duality. □

Our goal is thus to prove Lemma 3.4. Modulo a conditioning step we introduce later; this will be done by (a) fixing an A and proving that the inquality holds with a high enough probability, followed by (b) union bound over all possible A.

Let $Y_i = \mathbf{1}_{i \in S'}$. For convenience, let us denote the max{} terms in equations (3.8) and (3.10) by u_i and $\tilde{u}_i^*$, respectively. Now,

$$\mathsf{Dual}_H^* - \frac{1}{\Delta^2}\mathsf{Dual}_G = \sum_i \left(Y_i \widetilde{\rho}_i^* z_i - \frac{1}{\Delta^2} \cdot \rho_i z_i \right) + \left(Y_i \tilde{u}_i^* - \frac{1}{\Delta^2} \cdot u_i \right). \tag{3.15}$$

We view the RHS as two summations (shown by the parentheses), and bound them separately.

The first is relatively easy. Recall that by definition, $\widetilde{\rho}_i^* = \frac{\rho_i}{p_i^* \Delta^2}$. Thus the first term is equal to $\sum_i \frac{\rho_i z_i}{p_i^* \Delta^2} (Y_i - p_i^*)$. The expectation of this quantity is 0. We will apply Bernstein's inequality to bound its magnitude. For this, note that the variance is at most (using $|z_i| \le 1$)

$$\sum_i \frac{\rho_i^2}{(p_i^*)^2 \Delta^4} p_i^* (1 - p_i^*) \le \sum_i \frac{4(d_i + \varepsilon\Delta)^2 (1 - p_i^*)}{p_i^* \Delta^4}.$$

The condition on S gives the bound on ρ_i that was used above. Next, we note that unless $p_i^* = 1$, we have $p_i^* \ge \frac{(d_i + \varepsilon\Delta)}{\alpha_\varepsilon \Delta^2}$. Thus the variance is bounded by $\sum_i \frac{4\alpha_\varepsilon \cdot (d_i + \varepsilon\Delta)}{\Delta^2} \le \frac{8\alpha_\varepsilon n}{\Delta}$. Next,

$$\max_i \left| \frac{\rho_i z_i}{p_i^* \Delta^2} \right| \le \frac{2(d_i + \varepsilon\Delta)}{p_i^* \Delta^2} \le 2\alpha_\varepsilon.$$

(Again, this is because we can ignore terms with $p_i^* = 1$, and for the rest, we have a lower bound.) Thus, by Bernstein's inequality,

$$\mathsf{Pr}[\left| \sum_i \frac{\rho_i z_i}{p_i^* \Delta^2} (Y_i - p_i^*) \right| \ge t] \le \exp\left(-\frac{t^2}{\frac{16\alpha_\varepsilon n}{\Delta} + 2t\alpha_\varepsilon} \right).$$

Setting $t = \varepsilon n / 4\Delta$, the bound simplifies to $\exp(-\frac{\varepsilon^2 n}{C \Delta \alpha_\varepsilon})$, for a constant C. Thus, by our choice of α_ε and our size bound on $|S|$, this is $< \exp(-|S|)/n^4$.

The second term of (3.15) requires most of the work. We start with the trick (which turns out to be important) of splitting it into two terms by adding a "hybrid" term, as follows:

$$\sum_i Y_i \tilde{u}_i^* - \frac{1}{\Delta^2} \cdot u_i = \sum_i \left(Y_i \tilde{u}_i^* - Y_i \frac{u_i}{p_i^* \Delta^2} \right) + \sum_i \left(Y_i \frac{u_i}{p_i^* \Delta^2} - \frac{1}{\Delta^2} \cdot u_i \right).$$

The second term will again be bounded using Bernstein's inequality (in which we use our earlier observation that $u_i = O(d_i)$). This gives an upper bound of $\varepsilon n / 8\Delta$, with probability $1 - \exp(-|S|)/n^4$. We omit the easy details.

Let us focus on the first term. We now use the simple observation that $\max\{0, A\} - \max\{0, B\} \leq |A - B|$, to bound it by

$$\sum_i Y_i \left| \widetilde{d}_i^* - \widetilde{\rho}_i^* - \sum_{j \in \Gamma(i) \cap S'} w_{ij}^* z_j - \frac{1}{p_i^* \Delta^2} \big(d_i - \rho_i - \sum_{j \in \Gamma(i)} z_j \big) \right| .$$

By the definition of $\widetilde{\rho}_i^*$, it cancels out. Now, writing $c_j = 1 - z_j$ (which now $\in [0, 2]$) and using the definition of $\widetilde{d}_i^*$, we can bound the above by

$$\sum_i Y_i \left| \sum_{j \in \Gamma(i)} Y_j w_{ij}^* c_j - \frac{1}{p_i^* \Delta^2} c_j \right| = \sum_i Y_i \left| \sum_{j \in \Gamma(i)} w_{ij}^* c_j (Y_j - p_j^*) \right| \qquad (\text{using } w_{ij}^* = \frac{1}{p_i^* p_j^* \Delta^2})$$

Showing a concentration bound for such a quadratic function will be subject of the rest of the section. Let us define

$$f(Y) := f(Y_1, \ldots, Y_n) := \sum_i Y_i \left| \sum_{j \in \Gamma(i)} w_{ij}^* c_j (Y_j - p_j^*) \right| . \tag{3.16}$$

We wish to show that $\mathbf{Pr}[f > \frac{\varepsilon n}{\Delta}] \leq \exp(-|S|)$. Unfortunately, this is not true – there are counter-examples (in which the neighborhoods of vertices have significant overlaps) for which it is not possible to obtain a tail bound better than $\exp(-n/\Delta^2)$, roughly speaking. To remedy this, we resort to a trick developed in [77, 88]. The key idea is to *condition* on the event that vertices have a small weighted degree into the set S', and obtain a stronger tail bound.

3.5.1 "Good" Conditioning

We say that a choice of Y's is *good* if for all $i \in V$, we have

$$\sum_{j \in \Gamma(i)} w_{ij}^* Y_j \leq \frac{\varepsilon \Delta + 2 d_i}{p_i^* \Delta^2} .$$

The first lemma is the following.

Lemma 3.5. *Let H be the weighted graph on S' obtained by our algorithm. For any vertex $i \in V$, we have*

$$\mathbf{Pr}\left[\sum_{j \in \Gamma(i)} w_{ij}^* Y_j > \frac{\varepsilon \Delta + 2 d_i}{p_i^* \Delta^2} \right] < \frac{1}{n^6} .$$

Proof. Fix some $i \in V$, and consider $\sum_{j\in\Gamma(i)} w^*_{ij} Y_j = \frac{1}{p^*_i \Delta^2}\left(\sum_{j\in\Gamma(i)} \frac{Y_j}{p^*_j}\right)$. The term in the parenthesis has expectation precisely d_i. Thus, applying Bernstein using $\max_j \frac{1}{p^*_j} \leq \frac{\alpha_\varepsilon \Delta}{\varepsilon}$, together with $\sum_{j\in\Gamma(i)} \frac{p^*_j(1-p^*_j)}{(p^*_j)^2} \leq d_i \max_j \frac{1}{p^*_j}$, we have

$$\mathbf{Pr}[\sum_{j\in\Gamma(i)\cap V_H} \frac{Y_j}{p^*_j} > d_i + t] \leq \exp\left(-\frac{\varepsilon t^2}{(d_i+t)\alpha_\varepsilon \Delta}\right).$$

By setting $t = (d_i + \varepsilon\Delta)$, the RHS above can be bounded by

$$\exp\left(-\frac{\varepsilon(d_i+\varepsilon\Delta)^2}{(2d_i+\varepsilon\Delta)\alpha_\varepsilon\Delta}\right) \leq \exp\left(-\frac{\varepsilon^2}{2\alpha_\varepsilon}\right) < \frac{1}{n^6}.$$

This completes the proof, using our choice of α_ε. □

Conditioning on the Y being good, we show the following concentration theorem.

Theorem 3.13. *Let Y_i's be independent random variables, that are* 1 *w.p. p^*_i and* 0 *otherwise, and let $f(Y)$ be defined as in* (3.16). *Then we have*

$$\boldsymbol{Pr}[f(Y) \geq \frac{\varepsilon n}{8\Delta} \mid Y \textit{ is good}] \leq \frac{1}{n^5} \cdot e^{-20n\log n/\varepsilon^2}.$$

We observe that the theorem implies Lemma 3.4. This is because by the Theorem and the proceeding discussions, $\mathbf{Pr}[\mathsf{Dual}^*_H - \mathsf{Dual}_G \leq \varepsilon n/\Delta \mid Y \text{ good}] \geq 1 - \frac{\exp(-|S|)}{n^5}$, for any $A \subseteq S$. Thus by union bound over A, $\mathbf{Pr}[\max_A \mathsf{Dual}^*_H - \max_A \mathsf{Dual}_G \leq \varepsilon n/\Delta \mid Y \text{ good}] \geq 1 - \frac{1}{n^5}$. Since the probability of the good event is at least $1 - \frac{1}{n^5}$ (by Lemma 3.5), the desired conclusion follows.

3.5.2 Concentration Bound for Quadratic Functions

To conclude our proof, it suffices to show Theorem 3.13. To bound the quadratic function f, we bound the moment generating function (MGF), $\mathbf{E}[e^{\lambda f} \mid \text{good}]$. This is done via a decoupling argument, a standard tool for dealing with quadratic functions. While decoupling is immediate for 'standard' quadratic forms, the proof also works for our f (which has additional absolute values). The rest of the proof has the following outline.

3.5.3 Proof Outline

The main challenge is the computation of the MGF under conditioning (which introduces dependencies among the Y_i, albeit mild ones). The decoupling allows us to partition vertices into two sets, and only consider edges that go across the sets. We then show that it suffices to bound the MGF under a "weakened" notion of conditioning (a property we call δ-good). Under this condition, all the vertices in one of the sets of the partition become independent, thus allowing a bound on the moment — in terms of quantities that depend on the variables on the other set of the partition. Finally, we appeal to a strong concentration bound of Boucheron et al. [38] to obtain an overall bound, completing the proof.

Here we present the details of the proof.

3.5.4 Decoupling

Consider independent Bernoulli random variables δ_i that take values 0 and 1 w.p. $1/2$ each, and consider the function

$$f_\delta := \sum_i \delta_i Y_i | \sum_{j \in \Gamma(i)} (1-\delta_j) w^*_{ij} c_j (Y_j - p^*_j)|$$

Using the fact that $\mathbb{E}[|g(x)|] \geq |\mathbb{E}[g(x)]|$ for any function g, and defining $\mathbb{E}_\delta$ as the expectation with respect to the δ_i's, we have

$$\begin{aligned}
\mathbf{E}_\delta f_\delta &= \mathbf{E}_\delta \sum_i \delta_i Y_i | \sum_{j \in \Gamma(i)} (1-\delta_j) w^*_{ij} c_j (Y_j - p^*_j)| = \sum_i \frac{1}{2} \cdot Y_i \mathbf{E}_\delta | \sum_{j \in \Gamma(i)} (1-\delta_j) w^*_{ij} c_j (Y_j - p^*_j)| \\
&\geq \sum_i \frac{1}{2} \cdot Y_i | \sum_{j \in \Gamma(i)} \mathbf{E}_\delta (1-\delta_j) w^*_{ij} c_j (Y_j - p^*_j)| \\
&= \sum_i \frac{1}{4} \cdot Y_i | \sum_{j \in \Gamma(i)} w^*_{ij} c_j (Y_j - p^*_j)| = \frac{1}{4} f
\end{aligned}$$

We used the fact that i never appears in the summation term involving Y_i to obtain the first equality. Next, using Jensen's inequality, we have:

$$\mathbf{E}_Y[e^{\lambda f} \mid \text{good}] \leq \mathbf{E}_Y[e^{4\lambda \mathbf{E}_\delta f_\delta} \mid \text{good}] \leq \mathbb{E}_{Y,\delta}[e^{4\lambda f_\delta} \mid \text{good}]$$

where $\mathbb{E}_{Y,\delta}$ means the expectation with respect to both random variables Y and δ. Now, the *interpretation* of f_δ is simply the following. Consider the partitioning (V^+, V^-) of V defined by $V^+ = \{i \in [n] : \delta_i = 1\}$ and $V^- = \{i \in [n] : \delta_i = 0\}$, then

$$f_\delta = \sum_{i \in V^+} Y_i | \sum_{j \in \Gamma(i) \cap V^-} w^*_{ij} c_j (Y_j - p^*_j)|.$$

For convenience, define $R_i = |\sum_{j \in \Gamma(i) \cap V^-} w^*_{ij} c_j (Y_j - p^*_j)|$, for $i \in V^+$. Thus we can write $f_\delta = \sum_{i \in V^+} Y_i R_i$. The condition that Y is good now gives us a bound on R_i. For any c_j (it is important to note that the good condition does not involve the constants c_j, as those depend on the LP solution; all we know is that $0 \le c_j \le 2$), we have

$$\begin{aligned} R_i &\le \Big| \sum_{j \in \Gamma(i) \cap V^-} 2w^*_{ij}(Y_j + p^*_j) \Big| \\ &\le \frac{2(\varepsilon\Delta + 2d_i)}{p^*_i \Delta^2} + \frac{2d_i}{p^*_i \Delta^2} \le \frac{2\varepsilon\Delta + 6d_i}{p^*_i \Delta^2}. \end{aligned}$$

Now, the quantity we wish to bound can be written as

$$\mathop{\mathbf{E}}_{Y}[e^{\lambda f} \mid \text{good}] \le \mathop{\mathbf{E}}_{Y,\delta}[e^{4\lambda f_\delta} \mid \text{good}] \le \mathop{\mathbf{E}}_{\delta} \mathop{\mathbf{E}}_{Y^-} \mathop{\mathbf{E}}_{Y^+} [e^{4\lambda \sum_{i \in V^+} Y_i R_i} \mid \text{good}]. \tag{3.17}$$

The key advantage that decoupling gives us is that we can now *integrate over* $Y_i \in V^+$, i.e., evaluate the innermost expectation, for any given choice of $\{Y_i : i \in V^-\}$ (which define the R_i). The problem with doing this in our case is that the good condition introduces dependencies on the Y_i, for $i \in V^+$.

Fortunately, weakening conditioning does not hurt much in computing expectations. This is captured by the following simple lemma.

Lemma 3.6. *Let Ω be a space with a probability measure μ. Let Q_1 and Q_2 be any two events such that $Q_1 \subset Q_2$, and let $Z : \Omega \mapsto \mathbb{R}^+$ be a* non-negative *random variable. Then,*

$$\boldsymbol{E}[Z|Q_1] \le \frac{\boldsymbol{E}[Z|Q_2]}{\boldsymbol{Pr}[Q_1]}.$$

Proof. Let Ω_1 (resp. Ω_2) be the subset of Ω in which Q_1 (resp. Q_2) is satisfied. By hypothesis, $\Omega_1 \subseteq \Omega_2$. Now by the definition of conditional expectation, and the non-negativity of Z, we have

$$\mathbf{E}[X|Q_1] = \frac{1}{\mu(Q_1)} \int_{x \in \Omega_1} Z(x)\mu(x)dx \le \frac{1}{\mu(Q_1)} \int_{x \in \Omega_2} Z(x)\mu(x)dx = \frac{\mu(Q_2)}{\mu(Q_1)} \mathbf{E}[Z|Q_2].$$

Since $\mu(Q_2) \le 1$, the conclusion follows. □

3.5.5 Weaker *Good* Property

The next crucial notion we define is a property "δ-good". Given a $\delta \in \{0,1\}^n$ (and corresponding partition (V^+, V^-)), a set of random variables Y is said to be δ-good if for all $i \in V^+$, we have

$$\sum_{j \in \Gamma(i) \cap V^-} w^*_{ij} Y_j \le \frac{\varepsilon\Delta + 2d_i}{p^*_i \Delta^2}. \tag{3.18}$$

We make two observations. First, the good property implies the δ-good property, for any choice of δ. Second, and more crucial to our proof, conditioning on δ-good does not introduce any dependencies on the variables $\{Y_i : i \in V^+\}$. Now, continuing from (3.17), and using the fact that the good condition holds with probability $> 1/2$, we have

$$\underset{\delta}{\mathbf{E}}\,\underset{Y^-}{\mathbf{E}}\,\underset{Y^+}{\mathbf{E}}\,[e^{4\lambda\sum_{i\in V^+} Y_i R_i} \mid \text{good}] \leq \underset{\delta}{\mathbf{E}}\,\underset{Y^-}{\mathbf{E}}\,\underset{Y^+}{\mathbf{E}}\,[2e^{4\lambda\sum_{i\in V^+} Y_i R_i} \mid \delta\text{-good}].$$

Now for any $0/1$ choices for variables Y^-, the R_i's get fixed for every $i \in V^+$, and we can bound $\mathbf{E}_{Y^+}[e^{4\lambda\sum_{i\in V^+} Y_i R_i} \mid R_i]$ easily.

Lemma 3.7. *Let Y_i be independent random $0/1$ variables taking value 1 w.p. p_i^*, and let R_i be given, for $i \in V^+$. Suppose $\lambda > 0$ satisfies $|\lambda R_i| \leq 1$ for all i. Then*

$$\mathbf{E}_{Y^+}[e^{\lambda\sum_{i\in V^+} Y_i R_i}] \leq e^{\sum_{i\in V^+} \lambda p_i^* R_i + \lambda^2 p_i^* R_i^2}.$$

Proof. Since the lemma only deals with $i \in V^+$, we drop the subscript for the summations and expectations. One simple fact we use is that for a random variable Z with $|Z| \leq 1$,

$$\mathbf{E}[e^Z] \leq \mathbf{E}[1 + Z + Z^2] \leq e^{\mathbf{E}[Z] + \mathbf{E}[Z^2]}.$$

Using this, and the independence of Y_i together with $Y_i^2 = Y_i$,

$$\mathbf{E}[e^{\lambda\sum Y_i R_i}] = \prod \mathbf{E}[e^{\lambda Y_i R_i}] \leq \prod e^{\mathbf{E}[\lambda Y_i R_i] + \mathbf{E}[\lambda^2 R_i^2 Y_i]}. \tag{3.19}$$

As $\mathbf{E}[Y_i] = p_i^*$, this completes the proof of the lemma. □

Using the lemma, replacing λ with 4λ yields the following

$$\underset{\delta}{\mathbf{E}}\,\underset{Y^-}{\mathbf{E}}\,\underset{Y^+}{\mathbf{E}}\,[e^{4\lambda\sum_{i\in V^+} Y_i R_i} \mid \delta\text{-good}] \leq \underset{\delta}{\mathbf{E}}\,\underset{Y^-}{\mathbf{E}}\,\left[e^{\sum_{i\in V^+} 4\lambda p_i^* R_i + 16\lambda^2 p_i^* R_i^2} \mid \delta\text{-good}\right]. \tag{3.20}$$

The second term in the summation is already small enough, i.e., using (3.18)

$$\sum_{i\in V^+} \lambda^2 p_i^* R_i^2 \leq 2\lambda^2 \alpha_\varepsilon \frac{n}{\Delta}.$$

While the bound on R_i can be used to bound the first term, it turns out that this is not good enough. We thus need a more involved argument. Thus the focus is now to bound

$$\underset{\delta}{\mathbf{E}}\,\underset{Y^-}{\mathbf{E}}\,\left[e^{\lambda g(Y)} \mid \delta\text{-good}\right], \text{ where } g(Y) := \sum_{i\in V^+} p_i^* \left| \sum_{j\in\Gamma(i)\cap V^-} w_{ij}^* c_j (Y_j - p_j^*) \right|.$$

3.5.6 Outline: Concentration Bound for g

To deduce a concentration bound for g, we first remove the conditioning (again appealing to Lemma 3.6). This then gives us independence for the Y_j, for $j \in V^-$. We can then appeal to the fact that changing a Y_j only changes g by a small amount, to argue concentration. However, the standard "bounded differences" concentration bound ([38]) will not suffice for our purpose, and we need more sophisticated results [38] (restated as Theorem 3.4).

To use the same notation as the theorem, define $Z = g(Y)$, where we only consider Y_r, $r \in V^-$. Now, for any such r, consider $Z - Z^{(r)}$. Since $Z^{(r)}$ is obtained by replacing Y_r by an independent Y'_r and recomputing g, we can see that the only terms i which could possibly be affected are $i \in \Gamma(r) \cap V^+$. Further, we can bound the difference $|Z - Z^{(r)}|$ by

$$|Z - Z^{(r)}| \le |Y_r - Y'_r| \sum_{i \in \Gamma(r) \cap V^+} 2p^*_i w^*_{ir},$$

where we have used $|c_r| \le 2$. The summation can be bounded by $\frac{2d_r}{p^*_r \Delta^2}$. Denote this quantity by θ_r. Then, to use the theorem, we need

$$\mathbf{E}_{Y'_r}|Z - Z^{(r)}|^2 \le |Y_r - Y'_r|^2 \theta_r^2 \le |Y_r - Y'_r| \theta_r^2 \le Y_r \theta_r^2 + p^*_r \theta_r^2.$$

We used the fact that $|Y_r - Y'_r| \in \{0, 1\}$. Now applying Theorem 3.4 by setting $\theta = \frac{1}{2\lambda}$, we have:

$$\mathbb{E}_{V^-}[e^{\lambda(g - \mathbf{E}[g])}] \le \mathbb{E}_{V^-}[e^{\frac{\lambda^2}{2} \sum_{r \in V^-} Y_r \theta_r^2 + p^*_r \theta_r^2}]$$

Once again, since $\lambda\theta_r$ will turn out to be < 1, we can use the bound $\mathbf{E}[e^{\lambda^2 Y_r \theta_r^2/2}] \le e^{\lambda^2 \theta_r^2 p^*_r}$, and conclude that

$$\mathbb{E}_{V^-}[e^{\lambda(g - \mathbf{E}[g])}] \le e^{2\lambda^2 \sum_{r \in V^-} p^*_r \theta_r^2} \le e^{4\lambda^2 \alpha_\varepsilon \frac{n}{\Delta}}.$$

The last inequality is due to a reasoning similar to earlier.

We are nearly done. The only step that remains for proving Theorem 3.13 is to obtain a bound on $\mathbf{E}[g]$. For this, we need to bound, for any i, the term

$$\mathbf{E}[|R_i|] = \mathbf{E}[| \sum_{j \in \Gamma(i) \cap V^-} w^*_{ij} c_j (Y_j - p^*_j)|].$$

By Cauchy-Schwartz and the fact that Y_j are independent, we have

$$\mathbf{E}[|R_i|]^2 \le \mathbf{E}[R_i^2] = \sum_{j \in \Gamma(i) \cap V^-} (w_{ij}^*)^2 c_j p_j^* (1 - p_j^*)$$
$$\le \frac{1}{(p_i^*)^2 \Delta^4} \sum_{j \in \Gamma(i)} \frac{1 - p_j^*}{p_j^*} \le \frac{\alpha_\varepsilon d_i}{\varepsilon (p_i^*)^2 \Delta^3}.$$

We have used the fact that $(1 - p_j^*)/p_j^* \le \alpha_\varepsilon \Delta / \varepsilon$. Thus we have

$$\mathbf{E}[\sum_{i \in V^+} p_i^* |R_i|] \le \left(\frac{\alpha_\varepsilon}{\varepsilon \Delta^3}\right)^{1/2} \sum_i d_i^{1/2} \le \left(\frac{\alpha_\varepsilon}{\varepsilon \Delta^3}\right)^{1/2} n \Delta^{1/2} \le \frac{n}{\Delta} \left(\frac{\alpha_\varepsilon}{\varepsilon}\right)^{1/2}.$$

From our choice of α_ε, this is $< \frac{1}{4} \cdot \frac{\varepsilon n}{\Delta}$.

Putting everything together, we get that the desired moment

$$\le \exp\left(\lambda \frac{\varepsilon n}{4\Delta} + \lambda^2 \alpha_\varepsilon \frac{n}{\Delta}\right).$$

To complete the bound, we end up setting $\lambda = \frac{\varepsilon}{\alpha_\varepsilon}$. For this value of λ, we must ensure that

$$\frac{\varepsilon}{\alpha_\varepsilon} \frac{(d_r + \varepsilon \Delta)}{p_r^* \Delta^2} \le 1,$$

which is indeed true.

3.5.7 Proof of Theorem 3.6 for MaxCut

First, observe that the expected size of the set S' is $\sum_i p_i \le \tilde{O}(n/\Delta)$. This can be shown to hold w.h.p. via simple Chernoff bounds.

Then, the concentration bound above implies Lemma 3.4. As we saw earlier, this completes the proof of Theorem 3.11. Now, following the technical outline presented in Section 3.3.1, this completes the proof of Theorem 3.6, for the case of MaxCut.

3.6 Sparse Coreset for Max-Cut

So far, we have shown that there is a core-set (i.e., a smaller weighted graph with the same MAXCUT value) with a small number of *vertices*. We now show that the number of edges can also be made small (roughly n/Δ). The result follows from the following lemma about sampling edges in graphs with edge weights ≤ 1. (We note that essentially the same lemma is used in several works on sparsifiers for cuts.)

Lemma 3.8. *Consider the weighted graph H resulted from* VERTEXSAMPLE, *defined on set of vertices S' in which $w_{ij} < 1$. If we apply* EDGESAMPLE*on H, then the resulted graph H' have*

$$\text{MaxCut}(H') \pm (1+\epsilon)\text{MaxCut}(H)$$

with probability at least $1 - \frac{1}{n^2}$.

Proof. Recall that in EDGESAMPLE algorithm we first rescale the edge weights so that they sum up to $|S'|$, and then sample each edge ij in the resulted graph (also denoted H, as we can assume it to be a preprocessing) with probability $p_{ij} = \min(w_{ij}\frac{8}{\varepsilon^2}, 1)$ and reweigh the edges to $w'_{ij} = \frac{w_{ij}}{p_{ij}}$ to obtain the graph H'. Define the indicator variable X_{ij} for each edge e_{ij} in graph H, where $X_{ij} = 1$ if the corresponding edge e_{ij} is selected by EDGESAMPLE and $X_{ij} = 0$ otherwise.

Our goal is to show that all cuts are preserved, w.h.p. Consider any cut (A, B) in H. Call the set of all the edges on this cut as $C_{A,B}$ and the set of in the cut on the sampled graph H' as $C'_{A,B}$. Set $w(C_{A,B}) = \sum_{e_{ij} \in C_{A,B}} w_{ij}$ and $w'(C'_{A,B}) = \sum_{e_{ij} \in C'_{A,B}} w'_{ij}$. Then we have,

$$\mathbb{E}[w'(C'_{A,B})] = \mathbb{E}[\sum w'_{ij} X_{ij}] = \sum w'_{ij} \mathbf{Pr}(X_{ij} = 1) = \sum p_{ij} w'_{ij} = \sum w_{ij} = w(C_{A,B}).$$

We will now apply Bernstein's inequality to bound the deviation. For this, the variance is first bounded as follows.

$$\mathbf{Var}[\sum w'_{ij} X_{ij}] = \sum {w'_{ij}}^2 \mathbf{Var}(X_{ij}) \leq \sum \frac{w_{ij}^2}{p_{ij}}(1 - p_{ij}) \leq \frac{\epsilon^2}{8} \sum w_{ij} = \frac{\epsilon^2}{8} w(C_{A,B})$$

We used the inequality that unless $p_{ij} = 1$ (in which case the term drops out), we have $w_{ij}/p_{ij} \leq \varepsilon^2/8$.

By the observation on w_{ij}/p_{ij} above, we can use Bernstein's inequality 3.3 with $b = \varepsilon^2/8$, to obtain

$$\mathbf{Pr}[|\sum w'_{ij} X_{ij} - w(C_{A,B})| \geq t] \leq \exp\left(-\frac{t^2}{\frac{\varepsilon^2 w(C_{A,B})}{4} + \frac{t\varepsilon^2}{8}}\right).$$

Setting $t = \varepsilon W$, where W is the sum of all the edge weights (which is equal to $|S'|$ after the preprocessing), the bound above simplifies to $\exp(-2|S'|)$, and thus we can take a union bound over all cuts. This completes the proof. □

3.6.1 Proof of Theorem 3.7

We only need to verify the bound on the number of edges. As every edge is sampled with probability $p_{ij} = \min(1, \frac{8w_{ij}}{\varepsilon^2})$, and since the total edge weight is normalized to be $|S'|$, we have that the expected number of edges is $\leq \frac{8|S'|}{\varepsilon^2}$, and w.p. at least $1 - \frac{1}{n^4}$, this is at most $\frac{16|S'|}{\varepsilon^2}$, completing the proof.

3.7 Correlation Clustering

We now show that precisely the same coreset construction used for MAXCUT also works for correlation clustering. Recall that we consider the MAX-AGREE version of correlation clustering. While correlation clustering is not a CSP (as the number of clusters is arbitrary), the following lemma shows that for the MAX-AGREE version, we can obtain a $(1 - \varepsilon)$ approximation, while restricting the number of clusters to $1/\varepsilon$.

Lemma 3.9. *Let $\mathcal{C}$ be the optimal clustering, and let OPT be its max-agree cost. Then there exists a clustering $\mathcal{C}'$ that has cost $\geq (1 - O(\varepsilon))OPT$, and has at most $1/\varepsilon$ clusters.*

The lemma is folklore in the correlation clustering literature. For the sake of completeness, we provide a simple proof here.

Proof. Let $A_1, A_2, \ldots, A_k$ be the clusters in the optimal clustering $\mathcal{C}$. Now, suppose we randomly color the clusters with $t = 1/\varepsilon$ colors, i.e., each cluster A_i is colored with a random color in $[t]$. We then merge all the clusters of a given color into one cluster, thus obtaining the clustering $\mathcal{C}'$. Clearly, $\mathcal{C}'$ has at most t colors.

Now, we observe that if $u, v \in A_i$ to begin with, then u, v are still in the same cluster in $\mathcal{C}'$. But if $u \in A_i$ and $v \in A_j$, and the colors of A_i and A_j are the same, then u and v are no longer separated in $\mathcal{C}'$. Let us use this to see what happens to the objective. Let χ_{uv} be an indicator for u, v being in the same cluster in the optimal clustering $\mathcal{C}$, and let χ'_{uv} be a similar indicator in $\mathcal{C}'$. The original objective is

$$C^- + \sum_{ij} \eta_{ij} \chi_{ij}.$$

From the above reasoning, if $\chi_{uv} = 1$, then $\chi'_{uv} = 1$. Also, if $\chi_{uv} = 0$, $\mathbf{E}[\chi'_{uv}] = 1/t$, i.e., there is a probability precisely $1/t = \varepsilon$ that the clusters containing u, v now get the same color. Thus, we can write the expected value of the new objective as

$$C^- + \sum_{ij} \eta_{ij} \chi_{ij} + \varepsilon \sum_{ij} \eta_{ij} (1 - \chi_{ij}).$$

Let us denote $S = \sum_{ij} \eta_{ij}\chi_{ij}$ and $S' = \sum_{ij} \eta_{ij}(1 - \chi_{ij})$. Then by definition, we have $S + S' = \sum_{ij} \eta_{ij} = C^+ - C^-$. Now, to show that we have a $(1 - \varepsilon)$ approximation, we need to show that $C^- + S + \varepsilon S' \geq (1 - \varepsilon)(C^- + S)$. This simplifies to $C^- + S + S' \geq 0$, or equivalently, $C^+ \geq 0$, which is clearly true. This completes the proof. □

Another easy observation about MAX AGREE is that the objective value is at least $\max\{C^+, C^-\} \geq n\Delta/2$. This is simply because placing all the vertices in a single cluster gives a value C^+, while placing them all in different clusters gives C^-. Thus, it suffices to focus on additive approximation of $\varepsilon n\Delta$.

Once we fix the number of clusters k, we can write correlation clustering as a quadratic program in a natural way: for each vertex i, have k variables $x_{i\ell}$, which is supposed to indicate if i is given the label ℓ. We then have the constraint that $\sum_\ell x_{i\ell} = 1$ for all i. The objective function then has a clean form:

$$\sum_{ij}[\sum_{\ell=1}^{k} x_{i\ell}(1 - x_{j\ell})c^-_{ij} + x_{i\ell}x_{j\ell}c^+_{ij}] = \sum_{ij}\sum_{\ell} x_{i\ell}c^-_{ij} + x_{i\ell}x_{j\ell}\eta_{ij} = \sum_{i,\ell} x_{i\ell}(\rho_{i\ell} + d^-_i),$$

where $x_{i\ell} = 1$ iff vertex $i \in C_\ell$, and $\rho_{i\ell} = \sum_{j\in\Gamma(i)} x_{j\ell}\eta_{ij}$ and $d^-_i = \sum_j c^-_{ij}$.

Note the similarity with the program for MAXCUT. We now show that the same framework we used to prove our coreset result for MAXCUT (outlined in Section 3.3.1) carries over with minor changes. Let us start with the analog for the AKK Estimation procedure of Section 3.4.

3.7.1 LP Estimation Procedure for Correlation Clustering

Let $H = (V, E, c)$ be a weighted, undirected graph with edge weights c^+_{ij} and c^-_{ij} as before, and let $\gamma : V \to [0, 1]$ denote sampling probabilities. We define AKK-Est$_C(H, \gamma)$ to be the output of the following randomized algorithm: sample a set S by including each vertex i in it w.p. γ_i independently; next, for each partition $(A_1, \cdots, A_k)$ of S, solve the LP defined below, and output the largest objective value.

$LP_{A_1,\cdots,A_k}(V)$ is the following linear program. (As before, we use constants $\rho_{i\ell} := \sum_{j\in\Gamma(i)\cap A_\ell} \frac{\eta_{ij}}{\gamma_j}$.). Once again, the best choice of $s_{i\ell}, t_{i\ell}$ for each pair i, ℓ are so that $s_{i\ell} + t_{i\ell} = |\rho_{i\ell} - \sum_{j\in\Gamma(i)} \eta_{ij}x_{j\ell}|$.

$$\begin{aligned} \text{maximize} \quad & \sum_{i\ell} x_{i\ell}(\rho_{i\ell} + d_i^-) - (s_{i\ell} + t_{i\ell}) \\ \text{subject to} \quad & \rho_{i\ell} - t_{i\ell} \le \sum_{j \in \Gamma(i)} \eta_{ij} x_{j\ell} \le \rho_{i\ell} + s_{i\ell} \quad \forall i, \ell \\ & \sum_{\ell} x_{i\ell} = 1 \quad \forall i \in [n] \\ & s_{i\ell}, t_{i\ell} \ge 0 \quad \forall i, \ell \end{aligned}$$

We now show a result analogous to the one earlier – that under appropriate conditions, AKK-Est$_C$ is approximately equal to the optimal correlation clustering objective value.

Theorem 3.14. *Let H be a weighted graph on n vertices with edge weights c_{ij}^+, c_{ij}^- that add up to W. Suppose the sampling probabilities γ_i satisfy the condition*

$$w_{ij} \le \frac{W\varepsilon^2}{8k^2 \log n} \frac{\gamma_i \gamma_j}{\sum_u \gamma_u} \quad \textit{for all } i, j. \tag{3.21}$$

Then, we have AKK-Est$_C(H, \gamma) \in CC(H) \pm \varepsilon W$*, with probability at least* $1 - 1/n^2$ *(where the probability is over the random choice of S).*

Note that the only difference is the k^2 term in (3.21).

Proof. As before, the proof follows from two complementary claims.

Claim 1. W.h.p. over the choice of S, there exists a partitioning $(A_1, \cdots, A_k)$ of S such that $LP_{A_1, \cdots, A_k}(H) \ge CC(H) - \varepsilon W$.

Claim 2. Consider any feasible solution to the LP above (for some values $\rho_{i\ell}, s_{i\ell}, t_{i\ell}$). There exists a partitioning in H of objective value at least the LP objective.

The proof of Claim 1 mimics the proof in the case of MAXCUT. We use Bernstein's inequality for every $\ell \in [k]$ with deviation being bounded by $\varepsilon(d_i + \Delta)/k$ in each term. This is why we need an extra k^2 term in the denominator of (3.21). We omit the details.

Proof of Claim 2. Suppose we have a feasible solution x to the LP of objective value $\sum_{i,\ell} x_{i\ell}(d_i^- + \rho_{i\ell}) - \sum_{i,\ell} |\rho_{i\ell} - \sum_{j \in \Gamma(i)} \eta_{ij} x_{j\ell}|$, and we wish to move to a partitioning of at least this value. To this end, define the quadratic form

$$Q(x) := \sum_{i,\ell} x_{i\ell}(d_i^- + \sum_{j \in \Gamma(i)} \eta_{ij} x_{j\ell}).$$

The first observation is that for any $x \in [0,1]^{nk}$, and any real numbers ρ_i, we have

$$Q(x) \geq \sum_{i,\ell} x_{i\ell}(d_i^- + \rho_{i\ell}) - \sum_{i,\ell} |\rho_{i\ell} - \sum_{j \in \Gamma(i)} \eta_{ij} x_{j\ell}|.$$

This is true simply because $Q(x) = \sum_{i,\ell} x_{i\ell}\big(d_i^- + \rho_{i\ell}\big) - x_{i\ell}\big(\rho_{i\ell} - \sum_{j\in\Gamma(i)} \eta_{ij}x_{j\ell}\big)$, and the fact that the second term is at least $-|\rho_{i\ell} - \sum_{j\in\Gamma(i)} \eta_{ij}x_{j\ell}|$, as $x_i \in [0,1]$.

Next, note that the maximum of the form $Q(x)$ over $[0,1]^{nk}$ has to occur at a boundary point, since for any fixing of variables other than the ith group of variables $x_{i1}, \cdots, x_{ik}$ for a given i, the form reduces to a linear function of $x_{i\ell}$, $1 \leq \ell \leq k$, which attains maximum at one of the boundaries when subject to the constraint $\sum_\ell x_{i\ell} = 1$. Using this observation repeatedly for $i \in [n]$ lets us conclude that there is a $y \in \{0,1\}^{nk}$ such that $Q(y) \geq Q(x)$. Since any such y corresponds to a partitioning, and $Q(y)$ corresponds to its objective value, the claim follows. □

This completes the proof of Theorem 3.14. □

As in the case of MAXCUT, we can show that the estimation procedure can be used for estimating both in the original graph (with uniform probabilities q_i), and with the graph H, with sampling probabilities q_i/p_i. We thus skip stating these claims formally.

3.7.2 Induced Linear Programs for Correlation Clustering

We next need to prove that the AKK-Est$_C$ procedures have approximately the same values on G and H (with appropriate γ's). To show this, we consider a sample (S, S') drawn as before, and show that

$$\max_{(A_1,\cdots,A_k):S} LP^{\gamma}_{A_1,\cdots,A_k}(V) \geq \Delta^2 \max_{(A_1,\cdots,A_k):S} LP^{\alpha}_{A_1,\cdots,A_k}(S') - \varepsilon n\Delta, \tag{3.22}$$

where $\gamma_i = q_i$ and $\alpha_i = q_i/p_i$. As before, we consider the duals of the two programs. This is the main place in which our correlation clustering analysis differs from the one for MAXCUT. The dual is as follows

$$\begin{aligned} \text{minimize} \quad & \sum_i u_i + \sum_{i,\ell} \rho_{i\ell} z_{i\ell} \\ \text{subject to} \quad & u_i + \sum_{j\in\Gamma(i)} \eta_{j\ell} z_{j\ell} \geq d_i^- + \rho_{i\ell} \quad \forall i, \ell \\ & -1 \leq z_{i\ell} \leq 1 \quad \forall i \in [n], \ell \in [k] \end{aligned}$$

The difference now is that for any vector z, the optimal choice of u_i is $\max_\ell\{d_i^- + \rho_{i\ell} - \sum_{j\in\Gamma(i)} \eta_{j\ell} z_{j\ell}\}$. This is now a maximum of k terms, as opposed to the max of 0 and one other term in the case of MAXCUT. But once again, we can think of the dual solution as being the vector z, and we again have $u_i \le 2d_i$. The dual of $LP^\alpha_{A_1,\cdots,A_k}(H)$ can be written down similarly.

As we did earlier, we take a solution z to the dual of the LP on G, and use the same values (for the vertices in S') as the solution to the dual on H. The objective values are now as follows.

$$\mathsf{Dual}_G = \sum_{i,\ell} \rho_{i\ell} z_{i\ell} + \sum_i \max_\ell \{d_i^- + \rho_{i\ell} - \sum_{j\in\Gamma(i)} \eta_{ij} z_{j\ell}\} \tag{3.23}$$

$$\mathsf{Dual}_H \le \sum_{i\in S',\ell} \widetilde{\rho}_{i\ell} z_{i\ell} + \sum_i \max_\ell \{\ \widetilde{d_i^-} + \widetilde{\rho}_{i\ell} - \sum_{j\in\Gamma(i)\cap S'} w_{ij}\eta_{ij} z_{j\ell}\} \tag{3.24}$$

Here also, we have $\widetilde{\rho}_{i\ell} = \frac{\rho_{i\ell}}{p_i\Delta^2}$. We now show that w.p. at least $1 - \frac{1}{n^4}$,

$$\max_{(A_1,\ldots,A_k):S} \mathsf{Dual}_H \le \frac{1}{\Delta^2} \max_{(A_1,\ldots,A_k):S} \mathsf{Dual}_G + \frac{\varepsilon n}{\Delta}. \tag{3.25}$$

We next use the same trick as in the MAXCUT case, and move to Dual^*_H, in which we use $\widetilde{\rho}^*_{i\ell} := \frac{\rho_{i\ell}}{p_i\Delta^2}$, weights w^*_{ij} as before. The proof of Lemma 3.3 applies verbatim. Thus it suffices to show that w.h.p. (assuming S satisfies conditions analogous to Lemma 3.4),

$$\max_{(A_1,\ldots,A_k):S} \mathsf{Dual}^*_H \le \frac{1}{\Delta^2} \max_{(A_1,\ldots,A_k):S} \mathsf{Dual}_G + \frac{\varepsilon n}{2\Delta}.$$

For this goal, consider the expression

$$\mathsf{Dual}^*_H - \frac{1}{\Delta^2}\mathsf{Dual}_G = \sum_i \left(\sum_\ell (Y_i \widetilde{\rho}^*_{i\ell} z_{i\ell} - \frac{1}{\Delta^2}\cdot \rho_{i\ell} z_{i\ell}) \right) + \left(Y_i \tilde{u}^*_i - \frac{1}{\Delta^2}\cdot u_i \right). \tag{3.26}$$

We view this as two summations (shown by the parentheses), and bound them separately. The first is relatively easy. We observe that by definition,

$$\widetilde{\rho}^*_{i\ell} = \sum_{j\in\Gamma(i)\cap C_\ell} \frac{w^*_{ij}\eta_{ij} p^*_j}{q_j} = \frac{\eta_{ij}}{p^*_i\Delta^2} \sum_{j\in\Gamma(i)\cap C_\ell} \frac{1}{q_i} = \frac{1}{p^*_i\Delta^2}\cdot \rho_{i\ell}. \tag{3.27}$$

This implies that we can write the first term as $\sum_{i\ell} \frac{\rho_{i\ell} z_{i\ell}}{p^*_i\Delta^2}(Y_i - p^*_i)$. This will then be bounded via Bernstein.

For bounding the second term, we start with the trick of splitting it into two terms by adding a "hybrid" term, as follows:

$$\sum_i Y_i \tilde{u}_i^* - \frac{1}{\Delta^2} \cdot u_i = \sum_i \left(Y_i \tilde{u}_i^* - Y_i \frac{u_i}{p_i^* \Delta^2} \right) + \sum_i \left(Y_i \frac{u_i}{p_i^* \Delta^2} - \frac{1}{\Delta^2} \cdot u_i \right).$$

The second term will again be bounded using Bernstein. We will need to use the fact that $u_i = O(d_i)$.

Let us thus consider the first term. We can appeal to the fact $|\max\{P_1, \dots, P_k\} - \max\{Q_1, \dots, Q_k\}| \le \sum_i |P_i - Q_i|$, to bound it by

$$\sum_i Y_i \sum_\ell \left| \widetilde{d}_i^{*-} + \widetilde{\rho}_{i\ell}^* - \sum_{j \in \Gamma(i) \cap S'} w_{ij}^* \eta_{ij} z_{j\ell} - \frac{1}{p_i^* \Delta^2} (d_i^- + \rho_{i\ell} - \sum_{j \in \Gamma(i)} \eta_{ij} z_{j\ell}) \right|.$$

Using (3.27) and $w_{ij}^* = \frac{1}{p_i^* p_j^* \Delta^2}$ we can bound the above by

$$\sum_i Y_i \sum_\ell \left| \sum_{j \in \Gamma(i): \eta_{ij} < 0} Y_j |\eta_{ij}| w_{ij}^* (1 - z_{j\ell}) - \frac{|\eta_{ij}|}{p_i^* \Delta^2} c_{j\ell} \right| + \sum_i Y_i \sum_\ell \left| \sum_{j \in \Gamma(i): \eta_{ij} > 0} |\eta_{ij}| w_{ij}^* z_{j\ell} (Y_j - p_j^*) \right|$$

$$= \sum_i Y_i \sum_\ell \left| \sum_{j \in \Gamma(i): \eta_{ij} < 0} |\eta_{ij}| w_{ij}^* (1 - z_{j\ell})(Y_j - p_j^*) \right| + \sum_i Y_i \sum_\ell \left| \sum_{j \in \Gamma(i): \eta_{ij} > 0} |\eta_{ij}| w_{ij}^* z_{j\ell} (Y_j - p_j^*) \right|$$

This leads to a sum over ℓ of terms of the form

$$\sum_i \left| \sum_{j \in \Gamma(i): \eta_{ij} \neq 0} |\eta_{ij}| w_{ij}^* (1 - z_{j\ell})(Y_j - p_j^*) \right| \tag{3.28}$$

The nice thing now is that we can appeal (in a black-box manner, using the boundedness of η and z) to the concentration bound for quadratic functions in Sections 3.5 and 3.5.2, with ε replaced by ε/k, to conclude the desired concentration inequality. For this goal, we again use a similar conditioning as for MAXCUT, which we state here.

Lemma 3.10. *Let H be the weighted graph obtained after sampling with probabilities p_i^*. For any vertex $i \in V$, we have*

$$\textbf{Pr}\left[\sum_{j \in \Gamma(i)} w_{ij}^* |\eta_{ij}| Y_j > \frac{\varepsilon \Delta + 2 d_i}{p_i^* \Delta^2} \right] < \frac{1}{n^4}.$$

Proof. Fix some $i \in V$, and consider $\sum_{j\in\Gamma(i)} w^*_{ij}|\eta_{ij}|Y_j = \frac{1}{p_i^*\Delta^2}\left(\sum_{j\in\Gamma(i)} \frac{Y_j|\eta_{ij}|}{p_j^*}\right)$. The term in the parenthesis has expectation precisely d_i. Thus, applying Bernstein using $\max_j \frac{1}{p_j^*} \le \frac{\alpha_\varepsilon\Delta}{\varepsilon}$, together with $\sum_{j\in\Gamma(i)} \frac{|\eta_{ij}|p_j^*(1-p_j^*)}{p_j^{*2}} \le d_i \max_j \frac{1}{p_j^*}$, we have

$$\mathsf{Pr}\Big[\sum_{j\in\Gamma(i)\cap V_H} \frac{|\eta_{ij}|Y_j}{p_j^*} > d_i + t\Big] \le \exp\left(-\frac{\varepsilon t^2}{(d_i+t)\alpha_\varepsilon\Delta}\right).$$

By setting $t = (d_i + \frac{\varepsilon\Delta}{k})$, the RHS above can be bounded by

$$\exp\left(-\frac{\varepsilon(d_i+\frac{\varepsilon\Delta}{k})^2}{(2d_i+\frac{\varepsilon\Delta}{k})\alpha_\varepsilon\Delta}\right) \le \exp\left(-\frac{\varepsilon^2}{2\alpha_\varepsilon}\right) < \frac{1}{n^4}.$$

This completes the proof, using our choice of α_ε. □

Conditioning on the Y being good, we can obtain the concentration bound for the quadratic function (3.28). This now lets us take a union bound over all possible partitions of S (of which there are at most k^n), to obtain (3.25), which then completes the proof of the main result (Theorem 3.6), for correlation clustering.

CHAPTER 4

PRELIMINARIES ON STREAMING VERIFICATION PROTOCOLS

There are two main models for streaming verification which we first discuss here. Then, we provide the main streaming verification protocols which we use in this dissertation.

4.1 Models for Streaming Verification

The two main models of streaming verification are streaming interactive proofs and annotated data streams.

4.1.1 Streaming Interactive Proof (SIP)

In this dissertation, we will mainly work in the *streaming interactive proof* (SIP) model first proposed by Cormode et al. [66]. In this model, there are two players, the prover P and the verifier V. The input consists of a *stream* τ of items from a universe $\mathcal{U}$. Let f be a function mapping τ to any finite set $\mathcal{S}$. A k-message SIP for f works as follows:

- V and P read the input stream and perform some computation on it.
- V and P then exchange k messages, after which V either outputs a value in $\mathcal{S} \cup \{\perp\}$, where $\perp$ denotes that V is not convinced that the prover followed the prescribed protocol.

V is randomized. There must exist a prover strategy that causes the verifier to output $f(\tau)$ with probability $1 - \varepsilon_c$ for some $\varepsilon_c \leq 1/3$. Similarly, for all prover strategies, V must output a value in $\{f(\tau), \perp\}$ with probability $1 - \varepsilon_s$ for some $\varepsilon_s \leq 1/3$. The values ε_c and ε_s are respectively referred to as the completeness and soundness errors of the protocol. The constant $1/3$ appearing in the completeness and soundness requirements is chosen by convention [21]. The constant $1/3$ can be replaced with any other constant in $(0, 1)$ without affecting the theory in any way. The protocols we design here will have perfect completeness ($\varepsilon_c = 0$).

4.1.2 Annotated Data Streams

The *annotated data streaming model* of Chakrabarti et al. [46] essentially corresponds to one-message SIPs. Technically, the annotated data streaming model allows the annotation to be interleaved with the stream updates, while the SIP model does not allow the prover and verifier to communicate until after the stream has passed. However, almost all known annotated data streaming protocols do not utilize the ability to interleave the annotation with the stream, and hence are actually 1-message SIPs. In other words, interaction here is restricted to a help message sent by the prover to the verifier after the data has passed. Here, the constraint is the total communication should be sublinear in terms of input size (so a few help messages are allowed; however the prover cannot for example replay the data stream a nonconstant number of times).

4.1.3 Protocol Costs

A streaming verification protocol has two costs: the verifier space, and the total communication, expressed as the number of bits exchanged between V and P. We will use the notation (A, B) to denote a SIP with verifier space $O(A)$ and total communication $O(B)$. We will also consider the number of rounds of communication between V and P.

4.2 Some Useful Verification Protocols

We will make use of some basic tools in our verification algorithms. We summarize the main properties of these protocols here: for more details, the reader is referred to the original papers.

4.2.1 Multi-Set Equality (MSE)

We are given streaming updates to the entries of two vectors $\mathbf{a}, \mathbf{a}' \in \mathbb{Z}^u$ and wish to check $\mathbf{a} = \mathbf{a}'$. Reed-Solomon fingerprinting is a standard technique to solve MSE using only logarithmic space.

Theorem 4.1 (MSE, [64]). *Suppose we are given stream updates to two vectors $\mathbf{a}, \mathbf{a}' \in \mathbb{Z}^u$ guaranteed to satisfy $|\mathbf{a}_i|, |\mathbf{a}'_i| \leq M$ at the end of the data stream. Let $t = \max(M, u)$. There is a streaming algorithm using $O(\log t)$ space, satisfying the following properties: (i) If $\mathbf{a} = \mathbf{a}'$, then the streaming algorithm outputs 1 with probability 1. (ii) If $\mathbf{a} \neq \mathbf{a}'$, then the streaming algorithm* *outputs 0 with probability at least* $1 - 1/t^2$.

Proof. Let $\mathbb{F}$ be a finite field of prime order, satisfying $6u^3 \leq |\mathbb{F}| \leq u^4$. We view each entry of $\mathbf{a}$ and $\mathbf{a}'$ as an element $\mathbb{F}$ in the natural way. At the start of the stream, the streaming algorithm picks an $\alpha \in \mathbb{F}$ at random, and computes $\mathsf{finger}(\mathbf{a}) = \sum_{i \in [i]} a_i \cdot \alpha^i$ and $\mathsf{finger}(\mathbf{a}') = \sum_{i \in [i]} a'_i \cdot \alpha^i$ with a single streaming pass over the input stream. The algorithm outputs 1 if and only if $\mathsf{finger}(\mathbf{a}) = \mathsf{finger}(\mathbf{a}')$. Property (i) clearly holds: if $\mathbf{a} = \mathbf{a}'$, then the algorithm outputs 1 with probability 1. To see that Property (ii) holds, observe that $\mathsf{finger}(\mathbf{a})$ is a univariate polynomial of degree at most u in the entries of $\mathbf{a}$, and similarly for $\mathsf{finger}(\mathbf{a}')$. If $\mathbf{a} \neq \mathbf{a}'$, these two polynomials are not equal. Property (ii) follows, because any two distinct polynomials of degree at most u over $\mathbb{F}$ can agree on at most u inputs. □

Here are the two other protocols that act as building blocks for our graph verification protocols.

4.2.2 Inverse Protocol (Finv)

Let $\mathbf{a} \in \mathbb{Z}^u$ be a (frequency) vector. The *inverse frequency* function F_k^{-1} for a fixed k is the number of elements of $\mathbf{a}$ that have frequency k: $F_k^{-1}(\mathbf{a}) = |\{i \mid \mathbf{a}_i = k\}|$. Let $h_k(i) = 1$ for $i = k$ and 0 otherwise. We can then define $F_k^{-1}(\mathbf{a}) = \sum_i h_k(\mathbf{a}_i)$. Note that the domain of h_k is $[M]$ where $M = \max_i \mathbf{a}_i$. We will refer to the problem of verifying a claimed value of F_k^{-1} as Finv. There is a simple SIP for Finv. We restate the related results here [66].

Lemma 4.1 (Finv, [66]). *Given stream updates to a vector* $\mathbf{a} \in \mathbb{Z}^u$ *such that* $\max_i \mathbf{a}_i = M$ *and a fixed integer k there is a SIP to verify the claim* $F_k^{-1}(\mathbf{a}) = K$ *with cost* $(\log^2 u, M \log^2 u)$ *in* $\log u$ *rounds.*

Proof. The proof is by a simple application of the sum-check protocol, which we discuss later in detail (c.f. Lemma 4.5). □

Note that the same result holds if instead of verifying an inverse query for a single frequency k, we wish to verify it for a set of frequencies. Let $S \subset [M]$ and let $F_S^{-1} = |\{i|\mathbf{a}_i \in S\}|$. Then using the same idea as above, there is a SIP for verifying a claimed value of F_S^{-1} with costs given by Lemma 4.1.

4.2.3 Subset Protocol

We now present a new protocol for a variant of the vector equality test described in Theorem 4.1. While this problem has been studied in the annotation model, it requires space-communication product of $\Omega(u^2)$ communication in that setting.

Lemma 4.2 (Subset)**.** *Let $E \subset [u]$ be a set of elements, and let $S \subset [u]$ be another set owned by P. There is a SIP to verify a claim that $S \subset E$ with cost $(\log^2 u, (|S| + \log u) \log u)$ in $\log u$ rounds.*

Proof. Consider a vector $\bar{\mathbf{a}}$ with length u, in which the verifier does the following updates: for each element in set E, increment the corresponding value in vector $\bar{\mathbf{a}}$ by $+1$ and for each element in set S, decrements the corresponding value in vector $\bar{\mathbf{a}}$ by -1. Let the vector $\mathbf{a} \in \{0, 1\}^u$ be the characteristic vector of E, and let $\mathbf{a}'$ be the characteristic vector of S. Thus, $\bar{\mathbf{a}} = \mathbf{a} - \mathbf{a}'$. By applying F_{-1}^{-1} protocol on $\bar{\mathbf{a}}$, verifier can determine if $S \subset E$ or not. Note that in vector $\bar{\mathbf{a}}$, $M = 1$. Then the protocol cost follows by Lemma 4.1. □

4.2.4 The PointQuery and RangeCount Protocols

An instance of the PointQuery problem consists of a stream of updates as described in previous protocols followed by a query $q \in [u]$. The goal is to compute the coordinate $\mathbf{a}_q$. Chakrabarti et al. [48] gave a 2-message SIP for the PointQuery problem with polylogarithmic cost.

Theorem 4.2 (Chakrabarti et al. [48])**.** *Suppose the input to PointQuery is guaranteed to satisfy $|\mathbf{a}_i| \leq m$ at the end of the data stream, where the bound m is known in advance. Then there is a two-round SIP for PointQuery on an input stream with length n, with space and communication costs both bounded by $O(\log u \log(m + \log u))$.*

Another important protocol from [48] that we will rely on solves the RangeCount problem. Let $(\mathcal{U}, \mathcal{R})$ be a range space. Let the input consist of a stream of elements from $\mathcal{U}$ followed by a range $R \in \mathcal{R}$. The goal is to verify a claim by P that $|R \cap \mathcal{U}| = k$. Chakrabarti et al. showed:

Theorem 4.3 (Chakrabarti et al. [48])**.** *There is a two-message SIP for RangeCount with space and communication cost bounded by $O(\log |\mathcal{R}| \log(|\mathcal{R}| \cdot n))$, where n is the length of the stream. In particular, for range spaces of bounded shatter dimension ρ, $\log |\mathcal{R}| = \rho \log n = O(\log n)$.*

4.2.5 The GKR Protocol

A standard approach to developing multiround interactive proofs is to verify properties of circuits that compute the desired function. One of the most powerful protocols of this form is due to Goldwasser et al. [89], and known as the GKR protocol. This remarkable protocol has polylogarithmic communication costs when applied to any circuit of polylogarithmic depth. The GKR protocol was adapted to the streaming setting by Cormode et al. [66], yielding the following result.

Lemma 4.3. *[66, 89] Let $\mathbb{F}$ be a finite field, and let $f \colon \mathbb{F}^u \to \mathbb{F}$ be a function of the entries of the frequency vector of a data stream (viewing the entries as elements of $\mathbb{F}$ field in the natural way). Suppose that f can be computed by an $O(\log(S) \cdot \log(|\mathbb{F}|))$-space uniform arithmetic circuit $\mathcal{C}$ (over $\mathbb{F}$) of fan-in 2, size S, and depth d, with the inputs of $\mathcal{C}$ being the entries of the frequency vector. Then, assuming that $|\mathbb{F}| = \Omega(d \cdot \log S)$, f possesses an SIP requiring $O(d \cdot \log S)$ rounds. The total space cost is $O(\log u \cdot \log |\mathbb{F}|)$ and the total communication cost is $O(d \cdot \log(S) \cdot \log |\mathbb{F}|)$.*

4.2.6 The Sum-Check Protocol

For completeness, we describe the technical details and analysis of sum-check protocol of Lund, Fortnow, Karloff, and Nisan [128] in the next section. Here, we just state the protocol itself in Figure 4.1, as well as its main properties and related complexity results.

The sum-check protocol satisfies perfect completeness, and has soundness error $\varepsilon \leq \deg(g)/|\mathbb{F}|$, where $\deg(g)$ denotes the total degree of g (see [128] for a proof). There is one round of prover–verifier interaction in the sum-check protocol for each of the v variables of g, and the total communication is $O(\deg(g))$ field elements.

Note that as described in Figure 4.1, the sum-check protocol assumes that the verifier has oracle access to g. However, this will not be the case in applications, as g will ultimately be a polynomial that depends on the input data stream. In order to apply the sum-check protocol in a streaming setting, it is necessary to assume that V can evaluate g at any point $\mathbf{r}$ in small space with a single streaming passover over the input (this assumption is made in Lemma 4.4). Alternatively, one can have the prover *tell* the verifier $g(\mathbf{r})$, and then prove to the verifier that the value $g(\mathbf{r})$ is as claimed, using further applications of the sum-check protocol, or heavier hammers such as the GKR protocol (cf. Section 4.2.5), which is itself based on the sum-check protocol.

Input: V is given oracle access to a v-variate polynomial g over finite field $\mathbb{F}$ and an $H \in \mathbb{F}$.
Goal: Determine whether $H = \sum_{(x_1,\ldots,x_v)\in\{0,1\}^v} g(x_1,\ldots,x_v)$.

- In the first round, P computes the univariate polynomial
$$g_1(X_1) := \sum_{x_2,\ldots,x_v\in\{0,1\}^{v-1}} g(X_1,x_2,\ldots,x_v),$$
and sends g_1 to V. V checks that g_1 is a univariate polynomial of degree at most $\deg_1(g)$, and that $H = g_1(0) + g_1(1)$, rejecting if not.
- V chooses a random element $r_1 \in \mathbb{F}$, and sends r_1 to P.
- In the jth round, for $1 < j < v$, P sends to V the univariate polynomial
$$g_j(X_j) = \sum_{(x_{j+1},\ldots,x_v)\in\{0,1\}^{v-j}} g(r_1,\ldots,r_{j-1},X_j,x_{j+1},\ldots,x_v).$$
V checks that g_j is a univariate polynomial of degree at most $\deg_j(g)$, and that $g_{j-1}(r_{j-1}) = g_j(0) + g_j(1)$, rejecting if not.
- V chooses a random element $r_j \in \mathbb{F}$, and sends r_j to P.
- In round v, P sends to V the univariate polynomial
$$g_v(X_v) = g(r_1,\ldots,r_{v-1},X_v).$$
V checks that g_v is a univariate polynomial of degree at most $\deg_v(g)$, rejecting if not.
- V chooses a random element $r_v \in \mathbb{F}$ and evaluates $g(r_1,\ldots,r_v)$ with a single oracle query to g. V checks that $g_v(r_v) = g(r_1,\ldots,r_v)$, rejecting if not.
- If V has not yet rejected, V halts and accepts.

Figure 4.1: Description of the sum-check protocol. $\deg_i(g)$ denotes the degree of g in the ith variable.

This protocol was first applied in the context of streaming algorithms by Cormode et al. [66]. The costs of this protocol are summarized in the following lemma.

Lemma 4.4. *Let g be a v-variate polynomial over $\mathbb{F}$, which may depend on an input stream τ. Denote the degree of g in variable i by $deg_i(g)$. Assume V can evaluate g at any point $\mathbf{r} \in \mathbb{F}$ with a streaming pass over τ, using $O(v \log |\mathbb{F}|)$ bits of space. There is an SIP for computing the function $F(\tau) = \sum_{\sigma \in \{0,1\}^v} g(\sigma)$. The total number of messages is $O(v)$ and the total communication $O(\sum_{i=1}^{v} \deg_i(g) \cdot \log |\mathbb{F}|)$. The space required by the verifier is $O(v \cdot \log |\mathbb{F}|)$.*

A simpler statement of sum-check protocol which we use for our graph verification algorithms is as follows:

Lemma 4.5 (Sum-Check, [66]). *Given streaming updates to a vector $\mathbf{a} \in \mathbb{Z}^u$ and a univariate polynomial $h \colon \mathbb{Z} \to \mathbb{Z}$, there is a SIP to verify that $\sum_{i \in [u]} h(\mathbf{a}_i) = K$ for some claimed K. The total number of rounds is $O(\log u)$ and the cost of the protocol is $(\log(u) \log |\mathbb{F}|, deg(h) \log(u) \log |\mathbb{F}|)$.*

4.3 Revisit the Sum-Check Protocol with Constant Rounds

As mentioned before, we have a $(\log u)$-rounds $(\log u)$-cost verification protocol for any frequency-based functions by applying the sum-check. In this section, we study the possibility of reducing the round-complexity of the sum-check protocol to constant-rounds.

For this goal, we revisit the sum-check protocol described in [66] and briefly explain the details of the protocol and the complexity analysis.

The sum-check which we present here is for verifying F_1 function on $\mathbf{a}$ defined as follows:

$$F_1(\mathbf{a}) = \sum_{i \in [u]} f_{\mathbf{a}}(i) = \sum_{x_1, \cdots, x_d \in [\ell]^d} f_{\mathbf{a}}(x_1, x_2, \cdots, x_d)$$

in which $u = [\ell]^d$. We can simply extend this protocol to any frequency-based function defined as $F(\mathbf{a}) = \sum_{i \in [u]} h(\mathbf{a}_i) = \sum_{i \in [u]} h \circ f(i)$.

We briefly describe the construction of this extension polynomial. Start from $f_{\mathbf{a}}$ and rearrange the frequency vector $\mathbf{a}$ into a d-dimensional array in which $u = \ell^d$ for a choosen parameter ℓ. This way we can write $i \in [u]$ as a vector $((i)_1^\ell, ..., (i)_d^\ell) \in [\ell]^d$. Now we pick a large prime number for field size $|\mathbb{F}| > u$ and define the *low-degree extension* (LDE) of $\mathbf{a}$ as $\tilde{f}_{\mathbf{a}}(\mathbf{x}) = \sum_{\mathbf{v} \in [\ell]^d} a_{\mathbf{v}} \chi_{\mathbf{v}}(\mathbf{x})$, in which $\chi_{\mathbf{v}}(\mathbf{x}) = \prod_{j=1}^{d} \chi_{v_j}(x_j)$ and $\chi_k(x_j)$ has this property that it is equal to 1 if $x_j = k$ and 0 otherwise. This indicator function can be defined by Lagrange basis polynomial as follows:

$$\frac{(x_j - 0)...(x_j - (k-1))(x_j - (k+1))...(x_j - (\ell - 1))}{(k-0)...(k-(k-1))(k-(k+1))...(k-(\ell-1))} \tag{4.1}$$

Observe that for any fixed value $\mathbf{r} \in [\mathbb{F}]^d$, $\tilde{f}_{\mathbf{a}}(\mathbf{r})$ is a linear function of $\mathbf{a}$ and can be evaluated by a streaming verifier as the updates arrive. This is the key to the implementation of the sum-check protocol with a streaming verifier.

At the start of protocol, before observing the stream, V picks a random point, presented as $\mathbf{r} = (r_1, \cdots, r_d) \in [\mathbb{F}]^d$ in the corresponding field. Then the verifier computes $\tilde{f}_{\mathbf{a}}(\mathbf{r})$ incrementally as it reads the stream updates on $\mathbf{a}$. After observing the stream, the verification protocol proceeds in d rounds as follows:

In the first round, P sends a polynomial $g_1(x_1)$, claimed as :

$$g_1(X_1) = \sum_{x_2, \cdots, x_d \in [\ell]^{d-1}} \tilde{f}_{\mathbf{a}}(X_1, x_2, \cdots, x_d)$$

Note that in this stage, if polynomial g_1 is the same as what is claimed here by P, then $F_1(\mathbf{a}) = \sum_{x_1 \in [\ell]} g_1(x_1)$.

Following this process, in round $j > 1$, V sends r_{j-1} to P. Then P sends a polynomial $g_j(x_j)$, claiming that:

$$g_j(X_j) = \sum_{x_{j+1}, \cdots, x_d \in [\ell]^{d-j}} \tilde{f}_{\mathbf{a}}(r_1, \cdots, r_{j-1}, X_j, x_{j+1}, \cdots, x_d)$$

In each round, V does consistency checks by comparing the two most recent polynomials as follows:

$$g_{j-1}(r_{j-1}) = \sum_{x_j \in [\ell]} g_j(x_j)$$

Finally, in the last round, P sends g_d which is claimed to be:

$$g_d(X_d) = \tilde{f}_{\mathbf{a}}(r_1, \cdots, r_{d-1}, X_d)$$

Now, V can check if $g_d(r_d) = \tilde{f}_{\mathbf{a}}(\mathbf{r})$. If this test (along with all the previous checks) passes, then V accepts and convinced that $F_1(\mathbf{a}) = \sum_{x_1 \in [\ell]} g_1(x_1)$.

4.3.1 Complexity Analysis

The protocol consists of d rounds, and in each of them a polynomial g_j is sent by P, which can be communicated using $O(\ell)$ words. This results in a total communication cost of $O(d\ell)$. V needs to maintain $\mathbf{r}$, $\tilde{f}_{\mathbf{a}}(\mathbf{r})$ which each requires $(d+1)$ words of space, as well as computing and maintaining the values for a constant number of polynomials in each round of sum-check. As described before, this is required for comparing the two most recent polynomials by checking

$$g_{j-1}(r_{j-1}) = \sum_{x_j \in [\ell]} g_j(x_j)$$

Each of the g_j communicated in round j is a univariate polynomial with degree $(\ell - 1)$ and can be described in $(\ell - 1)$ words. Let's represent each polynomial g_j as follows:

$$g_j(x_j) = \sum_{i \in [\ell-1]} c_{ij} x_j^i$$

In each round j the verifier requires to do the consistency checks over the recent polynomials as follows:

$$g_j(r_j) = \sum_{x_j \in [\ell]} \sum_{i \in [\ell-1]} c_{ij} x_j^i$$

By reversing the ordering over the sum operation, we can rewrite this check as:

$$g_{j-1}(r_{j-1}) = \sum_{i \in [\ell-1]} \sum_{x_j \in [\ell]} c_{ij} x_j^i = \sum_{i \in [\ell-1]} c_{ij} \sum_{x_j \in [\ell]} x_j^i$$

Let $y_i = \sum_{x_j \in [\ell]} x_j^i$. Then, this will be equivalent to:

$$g_{j-1}(r_{j-1}) = \sum_{i \in [\ell-1]} c_{ij} y_i$$

Both sides in this test can be computed and maintained in $O(1)$ words space as V reads the polynomials g_j presented by P in streaming manner. Thus, the total space required by V is $O(d)$ words.

By selecting ℓ as a constant (say 2), we obtain both space and communication cost $O(\log u)$ words for sum-check protocol which runs in $\log u$ rounds and the probablity of error is $\frac{\ell d}{|\mathbb{F}|}$.

Now to obtain constant-rounds protocol, we can set $\ell = O(u^{\frac{1}{\gamma}})$ for some integer constant $\gamma > 1$, and considering $u = [\ell]^d$, we get $d = \gamma = O(1)$ (note that d controls the the number of rounds), result in a protocol with constant rounds and total communication $O(u^{\frac{1}{\gamma}})$ words, while maintaining the low space cost $\gamma = O(1)$ words for V. The failure probability goes to $O(\frac{u^{\frac{1}{\gamma}}}{|\mathbb{F}|})$, which by choosing $|\mathbb{F}|$ larger than u^b it can be made less than $\frac{1}{u^b}$ for any constant b without changing the asymptotic bounds.

4.3.2 Constant Round for Frequency-Based Functions

Here for verifying any statistic $F(\mathbf{a}) = \sum_{i \in [u]} h(\mathbf{a}_i) = \sum_{i \in [u]} h \circ f(i)$ on frequency vector $\mathbf{a}$, we use similar ideas to basic sum-check protocol which we described for verifying F_1, but the polynomials communicated by prover will be based on functions $h \circ f_{\mathbf{a}}$:

In the first round, P sends a polynomial $g'_1(X_1)$, claimed as:

$$g'_1(X_1) = \sum_{x_2, \cdots, x_d \in [\ell]^{d-1}} h \circ f_{\mathbf{a}}(X_1, x_2, \cdots, x_d)$$

If polynomial g'_1 is the same as what is claimed here by P, then $F(\mathbf{a}) = \sum_{x_1 \in [\ell]} g'_1(x_1)$.

In each round $j > 1$, V sends r_{j-1} to P. Then P sends a polynomial $g'_j(X_j)$, claimed as:

$$g'_j(X_j) = \sum_{x_{j+1}, \cdots, x_d \in [\ell]^{d-j}} h \circ f_{\mathbf{a}}(r_1, \cdots, r_{j-1}, X_j, x_{j+1}, \cdots, x_d)$$

Again, consistency checks in each round is done by V by comparing the two most recent polynomials:

$$g'_{j-1}(r_{j-1}) = \sum_{x_j \in [\ell]} g'_j(x_j)$$

And finally the verification process will be completed in the last round by sending polynomial $g'_d(X_d)$ by P, claimed as:

$$g'_d(X_d) = h \circ f_{\mathbf{a}}(r_1, \cdots, r_{d-1}, X_d)$$

Followed by checking if $g'_d(r_d) = h \circ f_{\mathbf{a}}(\mathbf{r})$ by V.

4.3.3 Complexity Analysis

Protocol consists of d rounds, and in each of them a polynomial g'_j with degree $O(\deg(h) \cdot \ell)$ is sent by P, which can be communicated using $O(\deg(h) \cdot \ell)$ words. This results in a total communication cost of $O(\deg(h) \cdot d\ell)$. V needs to maintain $\mathbf{r}$, $h \circ f_{\mathbf{a}}(\mathbf{r})$ (each requires $O(d)$ words space) as well as computing and maintaining the value for a constant number of polynomials in streaming manner in each round of protocol (requires $O(1)$ words of space), which results in a total space of $O(d)$ words of space. By selecting ℓ as a constant (say 2), then we obtain communication cost $O(\deg(h) \cdot \log u)$ and space cost $O(\log u)$ words for sum-check protocol which runs in $\log u$ rounds and the probability of error is $\frac{\deg(h) \cdot \ell d}{|\mathbb{F}|}$.

Note that the number of variables over which the input polynomial to the sum-check protocol is defined determines the number of rounds and for any frequency-based function defined as $F(\mathbf{a}) = \sum_{i \in [u]} h(\mathbf{a}_i) = \sum_{i \in [u]} h \circ f(i)$, in which h is a univariate function, the number of variables will not change and will be the same as $f_{\mathbf{a}}$. This implies that by applying the same trick as described above for reducing the number of variables in $f_{\mathbf{a}}$ (by setting $\ell = O(u^{\frac{1}{\gamma}})$ for some integer constant $\gamma > 1$ and $d = \gamma = O(1)$), we can obtain a constant-round protocol for verifying any statistics $F(\mathbf{a})$ defined by the frequency vector on the input stream, with space cost $\gamma = O(1)$ words and communication cost $O(\deg(h) \cdot u^{\frac{1}{\gamma}})$ words, while keeping the probability of error as low as $O(\frac{\deg(h)}{u^b})$ for some integer $b > 1$ (by choosing $|\mathbb{F}| > u^b$).

Lemma 4.6 ([66]). *For any $\gamma < \log u$, there is a SIP to verify that $\sum_{i \in [u]} h(\mathbf{a}_i) = K$ for some claimed K, with γ rounds, space cost $\gamma \cdot \log u = O(\log u)$ bits and communication cost $O(\deg(h) \cdot u^{\frac{1}{\gamma}} \cdot \log u)$ bits, while keeping the probability of error as low as $O(\frac{\deg(h)}{u^b})$ for some integer $b > 1$ (by choosing $|\mathbb{F}| > u^b$).*

Corollary 4.1 ([66]). *Let h be a univariate polynomial defined on the frequency vector* **a** *under our model. For any $\gamma < \log u$, there is a SIP for verifying the function $F(\tau) = \sum_{i \in [u]} h(\mathbf{a}_i)$. The total number of rounds is γ and the cost of the protocol is $\left(\gamma \log u, u^{\frac{1}{\gamma}} \log u\right)$.*

CHAPTER 5

STREAMING VERIFICATION OF GRAPH PROPERTIES

In this chapter, we present streaming interactive proofs (SIPs) for graph problems.

5.1 Overview

We present streaming interactive proofs (SIPs) for graph problems that are traditionally hard for streaming, such as for the maximum matching problem (in bipartite and general graphs, both weighted and unweighted) as well for approximating the traveling salesperson problem. In particular, we present protocols that verify a matching *exactly* in a graph using polylogarithmic space and polylogarithmic communication apart from the matching itself. In all our results, we consider the input in the *dynamic streaming model*, where graph edges are presented in arbitrary order in a stream and we allow both deletion and insertion of edges. All our protocols use either $\log n$ rounds of communication or (if the output size is sufficiently large or we are willing to tolerate superlogarithmic communication) constant rounds of communication.

To prove the above results, we also need SIPs for subproblems like connectivity, minimum spanning tree and triangle counting. While it is possible to derive similar (and in some cases better) results for these subroutines using known techniques [89], we require explicit protocols that return structures that can be used in the computation pipeline for the TSP. Furthermore, our protocols for these problems are much simpler than what can be obtained by techniques in [89], which require some effort to obtain precise bounds on the size and depth of the circuits corresponding to more complicated parallel algorithms.

Our model is also different from a standard multipass streaming framework, since communication must remain sublinear in the input and in fact in all our protocols the verifier still reads the input exactly once.

From a technical perspective, our work continues the *sketching* paradigm for designing efficient graph algorithms. All our results proceed by building linear sketches of the input graph. The key difference is that our sketches are not approximate but algebraic: based on random evaluation of polynomials over finite fields. Our sketches use higher dimensional linearization ("tensorization") of the input, which might itself be of interest. They also compose: indeed, our solutions are based on building a number of simple primitives that we combine in different ways. Figure 5.1 illustrates the interconnections between our tools and results.

5.2 Related Work

There are several previous works on streaming verification and computation. Here we present a brief overview of the related work:

5.2.1 Outsourced Computation

Work on outsourced computation comes in three other flavors in addition to SIPs: firstly, there is work on reducing the verifier and prover complexity without necessarily making the verifier a sublinear algorithm [80, 89, 107], in some cases using cryptographic assumptions to achieve their bounds. Another approach is the idea of *rational proofs* [24, 55, 93, 94], in which the verifier uses a payment function to give the prover incentive to be honest. Moving to sublinear verifiers, there has been research on designing SIPs where the verifier runs in sublinear *time* [96, 142].

5.2.2 General Streaming Verification Algorithms

Chakrabarti et al. [46, 47] introduced the notion of *annotations* in data streams, whereby an all-powerful prover could provide annotations to a verifier in order to complete a stream computation. Cormode et al. [66] introduced the model of Streaming Interactive Proofs (SIPs), which extends the annotated data streaming model to allow for multiple rounds of interaction between the prover and verifier. They introduced a streaming variant of the classical sum-check protocol of Lund, Fortnow, Karloff, and Nisan [128], and used it to give logarithmic cost protocols for a variety of well-studied streaming problems. In subsequent works, protocols were developed in both models for graph problems and matrix-vector operations [64], were extended to deal with *sparse streams* (where the annotation size should be sublinear in the number of stream updates, rather than in the size of the data

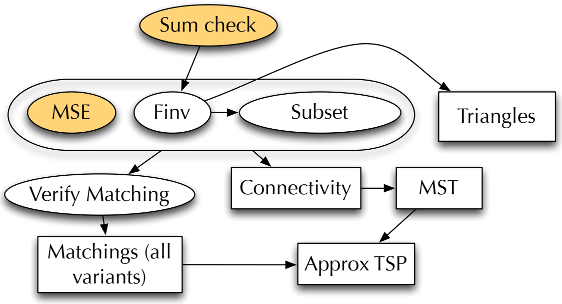


Figure 5.1: Summary of our tools and results. Subroutines are in ovals and problems are in rectangles. Shaded boxes indicate prior work. An arrow from A to B indicates that B uses A as a subroutine.

universe [44]), and were even shown to be implementable on GPUs [63]. Most recently, Chakrabarti et al. [48] developed streaming interactive proofs of logarithmic cost that worked in $O(1)$ rounds, making use of an interactive protocol for the Index problem.

Lower bounds on the cost of SIPs and their variants have also been studied [25, 44, 48, 118]. These results make use of *Arthur-Merlin communication complexity* and related notions. A detailed description of this line of work is outside the scope of this dissertation.

5.2.3 Streaming Graph Verification

All prior work on streaming graph verification has been in the annotation model, which in practice resembles a 1-round SIP (a single message from prover to verifier after the stream has been read). In recent work, Thaler [149] gives protocols for counting triangles, and computing maximum cardinality matching with both $n \log n$ space and communication cost. For matching, Chakrabarti et al. [45] show that any annotation protocol with space cost $O(n^{1-\delta})$ requires communication cost $\Omega(n^{1+\delta})$ for any $\delta > 0$. They also show that any annotation protocol for graph connectivity with space cost $O(n^{1-\delta})$ requires communication cost $\Omega(n^{1+\delta})$ for any $\delta > 0$.

It is also proved that every protocol for this problem in the annotation model requires $\Omega(n^2)$ product of space and communication. This is optimal up to logarithmic factors. Furthermore, they conjecture that achieving smooth tradeoffs between space and communication cost is impossible, i.e., it is not known how to reduce the space usage to $o(n \log n)$ without blowing the communication cost up to $\Omega(n^2)$ or vice versa [45, 149]. Note that in all our protocols, the product of space and communication is $O(n \operatorname{poly} \log n)$.

5.2.4 Streaming Graph Algorithms

In the general dynamic streaming model, poly $\log 1/\varepsilon$-pass streaming algorithms [11, 12] give $(1+\varepsilon)$-approximate answers and require $\tilde{O}(n)$ space. In one pass, the best results for matching are [59] (a parametrized algorithm for computing a maximal matching of size k using $\tilde{O}(nk)$ space) and [23, 123] which give a streaming algorithm for recovering an n^{ϵ}-approximate maximum matching by maintaining a linear sketch of size $\tilde{O}(n^{2-3\epsilon})$ bits. In the single-pass insert-only streaming model, Epstein et al. [74] give a constant (4.91) factor approximation for weighted graphs using $O(n \log n)$ space. Crouch and Stubbs [67] give a $(4+\epsilon)$-approximation algorithm which is the best known result for weighted matchings in this model. Triangle counting in streams has been studied extensively [32, 39, 42, 105, 138]. For dynamic graphs, the most space-efficient result is the one by [14] that provides the aforementioned additive εn^3 bound in polylogarithmic space. The recent breakthrough in sketch-based graph streaming [13] has yielded $\tilde{O}(n)$ semistreaming algorithms for computing the connectivity, bipartiteness and minimum spanning trees of dynamic graphs. For more details, see [132].

5.3 Overview of our Techniques

For all the problems that we discuss here the input is a data stream of edges of a graph drawn from $[n] \times [n]$ along with weight information as needed, where for an edge e an element in the stream is of the form (i, j, Δ). As is standard, we assume that edge weights are drawn from $[n^c]$ for some constant c. We allow edges to be inserted and deleted but the final edge multiplicity is 0 or 1, and also mandate that the length of the stream is polynomial in n. Finally, for weighted graphs, we further constrain that the edge weight updates be atomic, i.e., that an edge along with its full weight be inserted or deleted at each step. Now all our protocols proceed as follows. We define a domain $\mathcal{U}$ of size u and a frequency vector $\mathbf{a} \in \mathbb{Z}^u$ whose entries are indexed by elements of $\mathcal{U}$. A particular protocol might define a number of such vectors, each over a different domain. Each stream element will trigger a set of indices from $\mathcal{U}$ at which to update $\mathbf{a}$. For example in case of matching, we derive this constraint universe from the LP certificate, whereas for counting triangles our universe is derived from all $O(n^3)$ possible three-tuples of the vertices.

The key idea in all our protocols is that since we cannot maintain $\mathbf{a}$ explicitly due to limited space, we instead maintain a linear *sketch* of $\mathbf{a}$ that varies depending on the problem being solved. This sketch is computed as follows. We will design a polynomial that acts as a *low-degree* extension of f over an extension field $\mathbb{F}$ and can be written as $p(x_1, \ldots, x_d) = \sum_{u \in \mathcal{U}} a[u] g_u(x_1, x_2, \ldots, x_d)$. The crucial property of this polynomial is that it is *linear* in the entries of $\mathbf{a}$. This means that polynomial evaluation at any fixed point $\mathbf{r} = (r_1, r_2, \ldots, r_d)$ is easy in a stream: when we see an update $a[u] \leftarrow a[u] + \Delta$, we merely need to add the expression $\Delta g_u(\mathbf{r})$ to a running tally. Our sketch will always be a polynomial evaluation at a *random* point $\mathbf{r}$. Once the stream has passed, V and the prover P will engage in a conversation that might involve further sketches as well as further updates to the current sketch. In our descriptions, we will use the imprecise but convenient shorthand "increment $\mathbf{a}[u]$" to mean "update a linear sketch of some low-degree extension of a function of $\mathbf{a}$". It should be clear in each context what the specific function is.

As mentioned earlier, a single stream update of the form (i, j, Δ) might trigger updates in many entries of $\mathbf{a}$, each of which will be indexed by a multidimensional vector. We will use the wild-card symbol '$*$' to indicate that all values of that coordinate in the index should be considered. For example, suppose $\mathcal{U} \subseteq [n] \times [n] \times [n]$. The instruction "update $\mathbf{a}[(i, *, j)]$" should be read as "update all entries $\mathbf{a}[t]$ where $t \in \{(i, s, j) \mid s \in [n], (i, s, j) \in \mathcal{U}\}$". We show later how to do these updates implicitly, so that verifier time remains suitably bounded. Note that in the protocols presented in this chapter, we use the Finv protocol introduced in Chapter 3 but here the input to the Finv is not the graph edges themselves, but instead the Finv is applied to the derived stream updates triggered by each input stream element. As stated before, a single stream update of the form (i, j, Δ) might trigger updates in many entries of vector $\mathbf{a}$, which is defined based on the problem.

There are three parameters that control the complexity of our protocols: the vector length u, the length of stream s and the maximum size of a coordinate $M = max_i \mathbf{a}_i$. In the protocols discussed in this chapter, M will always be upper bounded by some polynomial in u, i.e. $\log M = O(\log u)$. All algorithms we present use linear sketches, and so the stream length s only affects verifier running time. In Lemma 5.6 we discuss how to reduce even this dependence, so that verifier update time becomes polylogarithmic on each step.

5.4 Warm-up: Counting Triangles

The number of triangles in a graph is the number of induced subgraphs isomorphic to K_3. Here we present a protocol to verify the number of triangles in a graph presented as a dynamic stream of edges. We will assume that at the end of the stream no edge has a net frequency greater than 1.

1. V processes the input data stream consisting of tuples (i, j, Δ) representing edges in the graph for F_3^{-1} with respect to a vector **a** indexed by entries from $\mathcal{U} = \{(i, j, k) \mid i, j, k \in [n], i < j < k\}$. For each edge $e = (i, j, \Delta), i < j$ in the stream, V increments all entries $\mathbf{a}[(i, *, j)], \mathbf{a}[(*, i, j)]$ and $\mathbf{a}[(i, j, *)]$ by Δ. Note that the input to F_3^{-1} protocol is in fact these derived incremental updates from the original stream of edges in the input graph and not the tuples (i, j, Δ).
2. P sends the claimed value c^* as the number of triangles in G.
3. V checks the the correctness of the answer by running the verification protocol for F^{-1} and checks if $F_3^{-1} = c^*$.

Lemma 5.1. *The above protocol correctly verifies (with a constant probability of error) the number of triangles in a graph with cost* $(\log^2 n, \log^2 n)$.

Proof. Follows from Lemma 4.1 and observation that maximum frequency of any entry in **a** is 3. □

5.5 SIP for MAX-MATCHING in Bipartite Graphs

We now present a SIP for maximum cardinality matching in bipartite graphs. The prover P needs to generate two certificates: an actual matching, and a proof that this is optimal. By König's theorem [122], a bipartite graph has a maximum matching of size k if and only if it has a minimum vertex cover of size k. Therefore, P's proof consists of two parts: i) Send the claimed optimal matching $M \subset E$ of size k. ii) Send a vertex cover $S \subset V$of size k. V has three tasks: i) Verify that M is a matching and that $M \subset E$. ii) Verify that S covers all edges in E. iii) Verify that $|M| = |S|$. We describe protocols for first two tasks and the third task is trivially solvable by counting the length of the streams and can be done in $\log n$ space. V will run the three protocols in parallel.

5.5.1 Verifying a Matching

Verifying that $M \subset E$ can be done by running the Subset protocol from Lemma 4.2 on E and the claimed matching M. A set of edges M is a matching if each vertex has degree at most 1 on the subgraph defined by M. Interpreted another way, let τ_M be the stream of endpoints of edges in M. Then each item in τ_M must have frequency 1. This motivates the following protocol, based on Theorem 4.1. V treats τ_M as a sequence of updates to a frequency vector $\mathbf{a} \in \mathbb{Z}^{|V|}$ counting the number of occurrences of each vertex. V then asks P to send a stream of all the vertices incident on edges of M as updates to a different frequency vector $\mathbf{a}'$. V then runs the MSE protocol to verify that these are the same.

5.5.2 Verifying that S is a Vertex Cover

The difficulty with verifying a vertex cover is that V no longer has streaming access to E. However, we can once again reformulate the verification in terms of frequency vectors. S is a vertex cover if and only if each edge of E is incident to some vertex in S. Let $\mathbf{a}, \mathbf{a}' \in \mathbb{Z}^{\binom{n}{2}}$ be vectors indexed by $\mathcal{U} = \{(i, j), i, j \in V, i < j\}$. On receiving the input stream edge $e = (i, j, \Delta), i < j$, V increments $\mathbf{a}[(i, j)]$ by Δ.

For each vertex $i \in S$ that P sends, we increment all entries $\mathbf{a}'[(i, *)]$ and $\mathbf{a}'[(*, i)]$. Now it is easy to see that S is a vertex cover if and only if there are no entries in $\mathbf{a} - \mathbf{a}'$ with value 1 (because these entries correspond to edges that have *not* been covered by a vertex in S). This yields the following verification protocol.

1. V processes the input edge stream for the F_1^{-1} protocol, maintaining updates to a vector $\mathbf{a}$.

2. P sends over a claimed vertex cover S of size c^* one vertex at a time. For each vertex $i \in S$, V *decrements* all entries $\mathbf{a}[(i, *)]$ and $\mathbf{a}[(*, i)]$.

3. V runs Finv to verify that $F_1^{-1}(\mathbf{a}) = 0$.

The bounds for this protocol follow from Lemmas 4.1, 4.2 and Theorem 4.1:

Theorem 5.1. *Given an input bipartite graph with n vertices, there exists a streaming interactive protocol for verifying the maximum-matching with* $\log n$ *rounds of communication, and cost* $(\log^2 n, (c^* + \log n) \log n)$*, where* c^* *is the size of the optimal matching.*

5.6 SIP for MAX-WEIGHT-MATCHING in Bipartite Graphs

Consider now a bipartite graph with edge weights, with the goal being to compute a matching of maximum weight (the weight of the matching being the sum of the weights of its edges). Our verification protocol will introduce another technique we call "flattening" that we will exploit subsequently for matching in general graphs.

Recall that we assume a "dynamic update" model for the streaming edges: each edge is presented in the form (e, w_e, Δ) where $\Delta \in \{+1, -1\}$. Thus, edges are inserted and deleted in the graph, but their weight is not modified. We will also assume that all weights are bounded by some polynomial n^c.

As before, one part of the protocol is the presentation of a matching by P: the verification of this matching follows the same procedure as in Section 5.5.1 and we will not discuss it further. We now focus on the problem of certifying *optimality* of this matching.

For this goal, we proceed by the standard LP-duality for bipartite maximum weight matching. Let the graph be $G = (V, E)$ and A is its incidence matrix (a matrix in $\{0,1\}^{V \times E}$ where $a_{ij} = 1$ iff edge j is incident to vertex i). Let $\delta(v)$ denote the edge neighborhood of a vertex v and P_{match} represent the convex combination of all matchings on G, and note that for a bipartite graph:

$$P_{\text{match}} = \left\{ x \in \mathbb{R}_+^E : \forall v \in V, \sum_{e \in \delta(v)} x_e \leq 1 \right\} \tag{5.1}$$

Applying the LP duality theorem to the bipartite max-weight matching problem on G, and letting w be the weight vector on the edges, we see that:

$$\begin{aligned}
\max\{w^T x : x \in P_{\text{match}}(G)\} &= \max \left\{ w^T x : x \geq 0 \text{ and } \forall v \in V, \sum_{e \in \delta(v)} x_e \leq 1 \right\} \\
&= \max \left\{ w^T x : x \geq 0, Ax \leq 1 \right\} \\
&= \min \left\{ 1^T y : A^T y \geq w, y \geq 0 \right\} \\
&= \min \left\{ 1^T y : y \geq 0 \text{ and } \forall e_{i,j} \in E, y_i + y_j \geq w_{i,j} \right\}
\end{aligned}$$

Considering this formulation, a certificate of optimality for a maximum weight matching of cost c^* is an assignment of weights y_i to vertices of V such that $\sum y_i = c^*$ and for each edge $e = (i, j), y_i + y_j \geq w_e$.

A protocol similar to the unweighted case would proceed as follows: P would send over a stream (i, y_i) of vertices, and the verifier would treat these as decrements to a vector over edges. V would then verify that no element of the vector had a value greater than zero. However, by Lemma 4.1, this would incur a communication cost linear in the maximum weight (since that is the maximum value of an element of this vector), which is prohibitively expensive.

The key is to observe that the communication cost of the protocol depends linearly on the maximum value of an element of the vector, but only *logarithmically* on the length of the vector itself. So if we can "flatten" the vector so that it becomes larger, but the maximum value of an element becomes smaller, we might obtain a cheaper protocol.

Let **a** be indexed by elements of $\mathcal{U} = \{((i,j), w, y_i, y_j) \mid (i,j) \in E, w, y_i, y_j \in [n^c], i < j, w \leq y_i + y_j\}$. $|\mathcal{U}| = O(n^{3c+2})$. The protocol proceeds as follows.

Intuitively, each entry of **a** corresponds to a valid dual constraint. When V reads the input stream of edges, it will increment counts for all entries of **a** that *could* be part of a valid dual constraint. Correspondingly, when P sends back the actual dual variables, V updates all compatible entries.

1. V processes the input edge stream for F_3^{-1} (with respect to **a**).
2. Upon seeing (e, w_e, Δ) in the stream, V accordingly updates all entries $\mathbf{a}[(e, w, *, *)]$ by Δ.
3. P sends a stream of (i, y_i) in increasing order of i.
4. V verifies that all $i \in [n]$ appear in the list. For each i, it increments all entries $\mathbf{a}[((i,*), *, y_i, *)]$ and $\mathbf{a}[(*, i), *, *, y_i)]$.
5. V verifies that $F_3^{-1}(\mathbf{a}) = m$ and accepts.

5.6.1 Protocol Correctness

Suppose the prover provides a valid dual certificate satisfying the conditions for optimality. Consider any edge $e = (i,j)$, the associated dual variables y_i, y_j and the entry $r = (e, w_{ij}, y_i, y_j)$. When e is first encountered, V will increment $a[r]$. When P sends y_i, r will satisfy the compatibility condition and $a[r]$ will be incremented. A similar increment will happen for y_j. Note that no other stream element will trigger an update of $a[r]$. Therefore, every satisfied constraint will yield an entry of **a** with value 3.

Conversely, suppose the constraint is not satisfied, i.e., $y_i + y_j < w_{ij}$. There is no corresponding entry of **a** to be updated in this case. This proves that the number of entries of **a** with value 3 is exactly the number of edges with satisfied dual constraints. The correctness of the protocol follows.

5.6.2 Protocol Complexity

The maximum frequency in **a** is at most 3 and the domain size $u = O(n^{3c+2})$. Note that this is in contrast with the representation first proposed that would have domain size n^2 and maximum frequency $O(n^c)$. In effect, we have *flattened* the representation. Invoking Lemma 4.1, as well as the bound for verifying the matching from Section 5.5, we obtain the following result.

Theorem 5.2. *Given a bipartite graph with n vertices and edge weights drawn from $[n^c]$ for some constant c, there exists a streaming interactive protocol for verifying the maximum-weight matching with $\log n$ rounds of communication, space cost $O(\log^2 n)$ and communication cost $O(n \log n)$.*

We can make a small improvement to Theorem 5.2. First note that the prover need only send the non-zero y_i in ascending order along with label to the verifier, who can implicitly assign $y_j = 0$ to all absent weights. This then reduces the communication to be linear in the *cardinality* and thereby also the cost of the maximum weight matching. Namely, we now have:

Theorem 5.3. *Given an input bipartite graph with n vertices and edge weights drawn from $[n^c]$ for some constant c, there exists a streaming interactive protocol for verifying the maximum-weight matching with $\log n$ rounds of communication, space cost $O(\log^2 n)$ and communication cost $O(c^* \log n)$, where c^* is the cardinality of the optimal matching over the input.*

Note that we assume that V knows the number of edges in the graph. This assumption can be dropped easily by merely summing over all updates Δ. Since we assume that every edge will have a final count of 1 or 0, this will correctly compute the number of edges at the end of the stream.

5.7 SIP for Maximum-Weight-Matching in General Graphs

We now turn to the most general setting: of maximum weight matching in general graphs. This of course subsumes the easier case of maximum cardinality matching in general graphs, and while there is a slightly simpler protocol for that problem based on the Tutte-Berge characterization of maximum cardinality matchings [35, 150], we will not discuss it here.

We will use the odd-set based LP-duality characterization of maximum weight matchings due to Cunningham and Marsh. Let $\mathbb{O}(V)$ denote the set of all odd-cardinality subsets of V Let $y_i \in [n^c]$ define non-negative integral weight on vertex v_i, $z_U \in [n^c]$ define a non-negative integral weight on an *odd-cardinality* subset $U \in \mathbb{O}(V)$, $w_{ij} \in [n^c]$ define the weight of an edge $e = (i, j)$ and $c^* \in [n^{c+1}]$ be the weight of a maximum weight matching on G. We define y and z to be *dual feasible* if $y_i + y_j + \sum_{\substack{U \in \mathbb{O}(V) \\ i,j \in U}} z_U \geq w_{i,j}, \forall i, j$

A collection of sets is said to be *laminar* if any two sets in the collection are either disjoint or nested (one is contained in the other). Note that such a family must have size linear in the size of the ground set. Standard LP-duality and the Cunningham-Marsh theorem state that:

Theorem 5.4 ([68]). *For every integral set of edge weights W, and choices of dual feasible integral vectors y and z, $c^* \leq \sum_{v \in V} y_v + \sum_{U \in \mathbb{O}(V)} z_U \left\lfloor \frac{1}{2}|U| \right\rfloor$. Furthermore, there exist vectors y and z that are dual feasible such that $\{U : z_U > 0\}$ is laminar and for which the above upper bound achieves equality.*

We design a protocol that will verify that each dual edge constraint is satisfied by the dual variables. The laminar family $\{U : z_U > 0\}$ can be viewed as a collection of nested subsets (each of which we call a *claw*) that are disjoint from each other. Within each claw, a set U can be described by giving each vertex v in order of increasing *level* $\ell(v)$: The number of sets in which v is contained (see Figure 5.2). The prover will describe a set U and its associated z_U by the tuple $(LI, \ell, r_U, \partial U)$, where $1 \leq LI \leq n$ is the index of the claw U is contained in, $\ell = \ell(U)$, $r_U = \sum_{U' \supseteq U'} z_{U'}$ and $\partial U = U \setminus \cup_{U'' \subset U} U''$. For an edge $e = (i, j)$ let $r_e = \sum_{i,j \in U, U \in \mathbb{O}(V)} z_U$ represent the weight assigned to an edge by weight vector z on the laminar family. Any edge whose endpoints lie in different claws will have $r_e = 0$. For a

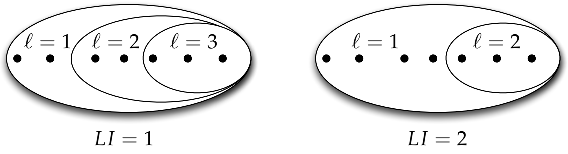


Figure 5.2: A laminar family

vertex v, let $r_v = \min_{v \in U} r_U$. For an edge $e = (v, w)$ whose endpoints lie in the same claw, it is easy to see that $r_e = \min(r_v, r_w)$, or equivalently that $r_e = r_{\arg\min(\ell(v),\ell(w))}$. For such an edge, let $\ell_{e,\downarrow} = \min(\ell(u), \ell(v))$ and $\ell_{e,\uparrow} = \max(\ell(u), \ell(v))$. We will use $LI(e) \in [n]$ to denote the index of the claw that the endpoints of e belong to.

5.7.1 The Protocol

V prepares to make updates to a vector $\mathbf{a}$ with entries indexed by $\mathcal{U} = \mathcal{U}_1 \cup \mathcal{U}_2$. $\mathcal{U}_1$ consists of all tuples of the form $\{(i, j, w, y, y', LI, \ell, \ell', r)\}$ and $\mathcal{U}_2$ consists of all tuples of the form $\{(i, j, w, y, y', 0, 0, 0, 0)\}$ where $i < j, i, j, LI, \ell, \ell' \in [n], y, y', r, w \in [n^c]$ and tuples in $\mathcal{U}_1$ must satisfy 1) $w \leq y + y' + r$ and 2) it is *not* simultaneously true that $y + y' \geq w$ and $r > 0$. Note that $\mathbf{a} \in \mathbb{Z}^u$ where $u = O(n^{4c+5})$ and all weights are bounded by n^c.

1. V prepares to process the stream for an F_5^{-1} query. When V sees an edge update of form (e, w_e, Δ), it updates all entries $\mathbf{a}[(e, w_e, *, *, *, *, *, *)]$.
2. P sends a list of vertices (i, y_i) in order of increasing i. For each (i, y_i), V increments by 1 the count of all entries $\mathbf{a}[(i, *, *, y_i, *, *, *, *, *]$ and $\mathbf{a}[(*, i, *, *, y_i, *, *, *, *)]$ with indices drawn from $\mathcal{U}_1$. Note that P only sends vertices with nonzero weight, but since they are sent in increasing order, V can infer the missing entries and issue updates to $\mathbf{a}$ as above. V also maintains the sum of all y_i.
3. P sends the description of the laminar family in the form of tuples $(LI, \ell, r_U, \partial U)$, sorted in lexicographic order by LI and then by ℓ. V performs the following operations.
    (a) V increments all entries of the form $(i, *, *, y_i, *, 0, 0, 0, 0)$ or $(*, i, *, *, y_i, 0, 0, 0, 0)$ by 2 to account for edges which are satisfied by only vector y.
    (b) V maintains the sum Σ_R of all r_U seen thus far. If the tuple is deepest level for a given claw (easily verified by retaining a one-tuple lookahead) then V adds r_U to another running sum $\Sigma_{\max}$.
    (c) V verifies that the entries appear in sorted order and that r_U is monotone

increasing.

(d) V updates the fingerprint structure from Theorem 4.1 with each vertex in ∂U.

(e) For each $v \in \partial U$, V increments (subject to our two constraints on the universe) all entries of $\mathbf{a}$ indexed by tuples of the form $(e, w_e, *, *, LI, *, \ell, *)$ and all entries indexed by tuples of the form $(e, w_e, *, *, LI, \ell, *, r_U)$, where e is any edge containing v as an endpoint.

(f) V ensures all sets presented are odd by verifying that for each LI, all $|\partial U|$ except the last one are even.

4. P sends V all vertices participating in the laminar family in ascending order of vertex label. V verifies that the fingerprint constructed from this stream matches the fingerprint constructed earlier, and hence that all the claws are disjoint.
5. V runs a verification protocol for $F_5^{-1}(\mathbf{a})$ and accepts if $F_5^{-1}(\mathbf{a}) = m$, returning Σ_r and $\Sigma_{\max}$.

Define c^s as the certificate size, which is upper bounded by the matching cardinality. Then:

Theorem 5.5. *Given dynamic updates to a weighted graph on n vertices with all weights bounded polynomially in n, there is a SIP with cost* $(\log^2 n, (c^s + \log n) \log n)$*, where* c^s *is the cardinality of maximum matching, that runs in* $\log n$ *rounds and verifies the size of a maximum weight matching.*

Proof. In parallel, V and P run protocols to verify a claimed matching as well as its optimality. The correctness and resource bounds for verifying the matching follow from Section 5.5. We now turn to verifying the optimality of this matching. The verifier must establish the following facts: (i) P provides a valid laminar family of odd sets. (ii) The lower and upper bounds are equal. (iii) All dual constraints are satisfied.

Since the verifier fingerprints the vertices in each claw and then asks P to replay all vertices that participate in the laminar structure, it can verify that no vertex is repeated and therefore that the family is indeed laminar. Each ∂U in a claw can be written as the difference of two odd sets, except the deepest one (for which $\partial U = U$). Therefore, the cardinality of each ∂U must be even, except for the deepest one. V verifies this claim, establishing that the laminar family is comprised of odd sets.

Consider the term $\sum_U z_U \lfloor |U|/2 \rfloor$ in the dual cost. Since each U is odd, this can be rewritten as $(1/2)(\sum_u z_u |U| - \sum_U z_U)$. Consider the odd sets $U_0 \supset U_1 \supset \ldots \supset U_l$ in a single claw. We have $r_{U_j} = \sum_{i \leq j} z_{U_i}$, and therefore $\sum_j r_{U_j} = \sum_j \sum_{i \leq j} z_{U_i}$. Reordering, this is equal to $\sum_{i \leq j} \sum_j z_{U_i} = \sum_i z_{U_i}|U_i|$. Also, $r_{U_l} = \sum_i z_{U_i}$. Summing over all claws, $\Sigma_r = \sum_U z_U|U|$ and $\Sigma_{\max} = \sum_U z_U$. Therefore, $\sum_i y_i + \Sigma_r - \Sigma_{\max}$ equals the cost of the dual solution provided by P.

Finally we turn to validating the dual constraints. Consider an edge $e = (i, j)$ whose dual constraints are satisfied, i.e., P provides y_i, y_j and z_U such that $y_i + y_j + r_{ij} \geq w_e$. Firstly, consider the case when $r_{ij} > 0$. In this case, the edge belongs to some claw LI. Let its lower and upper endpoints vertex levels be s, t, corresponding to odd sets U_S, U_t. Consider now the entry of **a** indexed by $(e, y_i, y_j, LI, s, t, r_{ij})$. This entry is updated when e is initially encountered and ends up with a net count of 1 at the end of input processing. It is incremented twice when P sends the (i, y_i) and (j, y_j). When P sends U_s this entry is incremented because $r_{ij} = r_{U_s} = \min(r_{U_S}, r_{U_t})$ and when P sends U_t this entry is incremented because U_t has level t, returning a final count of 5. If $r_{ij} = 0$ (for example when the edge crosses a claw), then the entry indexed by $(e, w_e, y_i, y_j, 0, 0, 0, 0)$ is incremented when e is read. It is not updated when P sends (i, y_i) or (j, y_j). When P sends the laminar family, V increments this entry by 2 twice (one for each of i and j) because we know that $y_i + y_j \geq w_e$. In this case, the entry indexed by $(i, j, w_e, y_i, y_j, 0, 0, 0, 0)$ will be exactly 5. Thus, for each satisfied edge there is exactly one entry of **a** that has a count of 5.

Conversely, suppose e is not satisfied by the dual constraints, for which a necessary condition is that $y_i + y_j < w_e$. Firstly, note that any entry indexed by $(i, j, w_e, *, *, 0, 0, 0, 0)$ will receive only two increments: one from reading the edge, and another from one of y_i and y_j, but not both. Secondly, consider any entry with an index of the form $(i, j, w_e, *, *, LI, *, *, *)$ for $LI > 0$. Each such entry gets a single increment from reading e and two increments when P sends (i, y_i) and (j, y_j). However, it will not receive an increment from the second of the two updates in Step 3(e), because $y_i + y_j + r_{ij} < w_e$ and so its final count will be at most 4. The complexity of the protocol follows from the complexity for Finv, Subset and the matching verification described in Section 5.5. □

5.8 Streaming Interactive Proofs for Approximate MST

For verifying the approximate weight of MST, we follow the reduction to the problem of counting the number of connected components in graphs, which was initially introduced in [53] and later was generalized to streaming setting [13]. Here are the main results which we use here:

Lemma 5.2 ([13]). *Let T be a minimum spanning tree on graph G with edge weights bounded by W =poly(n) and G_i be the subgraph of G consisting of all edges whose weights is at most $w_i = (1+\epsilon)^i$ and let $cc(H)$ denote the number of connected components of graph H. Set $r = \lfloor \log_{1+\epsilon} W \rfloor$. Then,*

$$w(T) \leq n - (1+\epsilon)^r + \sum_{i=0}^{r} \lambda_i cc(G_i) \leq (1+\epsilon) w(T)$$

where $\lambda_i = (1+\epsilon)^{i+1} - (1+\epsilon)^i$.

Based on this result, we can design a SIP for verifying the approximate weight of minimum spanning tree using a verification protocol 5.6 for number of connected components in a graph.

Theorem 5.6. *Given a weighted graph with n vertices, there exists a SIP protocol for verifying the number of connected components G_i with $(\log n)$ rounds of communication, and $(\log^2 n, n \log n)$ cost.*

Corollary 5.1. *Given a weighted graph with n vertices, there exists a SIP protocol for verifying MST within $(1+\epsilon)$-approximation with $(\log n)$ rounds of communication, and $(\log^2 n, n \log^2 n/\epsilon)$ cost.*

Proof. As the verifier processes the stream, each edge weight is snapped to the closest power of $(1+\epsilon)$. Note that given an a priori bound n^c on edge weights, G can be partitioned into at most $\frac{\log n}{\varepsilon}$ graphs G_i. We run this many copies of the connected components protocol in parallel to verify the values of $cc(G_i), \forall i$. □

We now present the proof of theorem 5.6. For simplicity, consider $V = (V_1 \cup \cdots \cup V_r)$ as the r connected components and $T = (T_i \cup \cdots \cup T_r)$ as r spanning trees on r corresponding connected components, provided by prover as the certificate. Now verifier needs to check if the certificate T is valid by considering the following conditions:

1. Disjointness: All the spanning trees are disjoint, i.e., $T_i \cap T_j = \emptyset$ for all the pairs of trees in T.
2. Subset: Each spanning tree in $T = (T_1 \cup \cdots \cup T_r)$ is a subgraph of the input graph G. This may be handled by Subset protocol described before in Lemma 4.2.
3. r-SpanningTree: Each component in $T = (T_1 \cup \cdots \cup T_r)$ is in fact a *spanning tree*.
4. Maximality: Each component in $T = (T_1 \cup \cdots \cup T_r)$ is in fact maximal, i.e., there is no edge between the components in original graph G.

We assume that the certificate T is sent by the prover in streaming manner in the following format and both players agree on this at the start of the protocol:

$$T : \{|T|, r, (T_1, \cdots, T_r)\}$$

For the representation of spanning trees, we consider a topological ordering on each tree T_i, starting from root node $root_i$, and each directed edge (v_{out}, v_{in}) connects the parent node v_{out} to the child node v_{in}:

$$T_i : \{root_i, \cup e(v_{out}, v_{in})\}$$

Here we present the protocols for checking each of these conditions. The following Disjointness protocol will be called a subroutine in our r-SpanningTree main protocol.

5.8.1 Protocol: Disjointness

1. P sends over r components of T_i in T in streaming manner.
2. P “replays" all the edges T' in the tuple form $(e_{i,j}, \ell)$ for $i, j \in [n]$ and $\ell \in [r]$ denotes the component $e_{i,j}$ is assigned to. The edges in T' are presented according to a canonical total ordering on the edge set, and hence V can easily check that T' has no repeated edge; i.e., the same edge presented in two distinct components.
3. Fingerprinting can then be used to confirm that $T' = \cup_i T_i$ with high probability, and hence that each edge occurs in at most one tree.
4. A similar procedure is run to ensure that no vertex is repeated in more than one T_i and that every vertex is seen at least once.

To check if each component in the claimed certificate T sent by the prover is in fact spanning tree, the verifier needs to check that each T_i is *cycle-free* and also *connected*. For this goal we present the following protocol:

5.8.2 Protocol: r-SpanningTree

1. P sends over the certificate $T : \{|T|, r, (T_1, \cdots, T_r)\}$ in which each T_i is of the form $\{i, \cup e(v_{out}, v_{in})\}$ and the root u_i of T_i is presented first.
2. V runs the Disjointness protocol to ensure that in the certificate $T : \{|T|, r, (T_1, \cdots, T_r)\}$ all T_i and T_j are edge and vertex disjoint for all $i \neq j$.
3. V has the prover again similarly replay $\cup_i T_i$ ordered by the label of the in-vertex of each edge to ensure that each vertex except the root u_i has exactly one incoming edge, i.e., that all T_i are cycle free and connected.

We now need to check that there is no edge between sets V_i and V_j for $i \neq j$:

5.8.3 Protocol: Maximality

1. We define an extended universe U now of size n^3, with elements $(e_{i,j}, k)$ where $k \in [n]$ represents the label of the component.
2. V initiates the F_{-1}^{-1} protocol on the input stream. Upon seeing any edge $e_{i,j}$, V increments by 1 all tuples containing $e_{i,j}$.
3. P sends the label of each vertex (v_i, j) where $j \in [n]$ represents the label of the component in the certificate. (Note that the verifier can ensure this input is consistent with the $T = \cup_i T_i$ sent earlier by simply fingerprinting as described before).
4. V considers each vertex $v_i \in V_j$ as a decrement update by 2 on all n possible tuples compatible with v_i and component label j. This step can be assumed as continuing the process for F_{-1}^{-1} mentioned in the first step.
5. F_{-1}^{-1} corresponds to exactly the set of edges observed in stream and crossing between two V_i and V_j for $i \neq j$. To see this, we enumerate the cases explicitly:
   (a) $\{-3, -4\}$ are the possible values for an edge $(e_{a,b}, i)$ with both endpoints contained in a single V_i corresponding to whether $e_{a,b}$ was originally in the stream or not. (The edge is decremented twice by 2 in the derived stream.)
   (b) $\{-1, -2\}$ are the possible values for any edge $(e_{a,b}, i)$ and $(e_{a,b}, j)$ with one endpoint in a V_i and the other in V_j for $i \neq j$, corresponding to whether $e_{a,b}$ was originally in the stream or not. ($e_{a,b}$ is decremented exactly once by 2 in each of the two copies corresponding to i and j, respectively.)
6. V runs F^{-1} with P and accepts that there are no edges between the V_i if and only if

$F_{-1}^{-1} = 0$.

5.8.4 Complexity Analysis of the Protocol

We know the cost of F_{-1}^{-1} protocol is $(\log^2 n, \log^2 n)$ for frequency ranges bounded by a constant, whereas the costs of the remaining fingerprinting steps and sending the certificate are at most $(\log n, n \log n)$. Hence the cost of our protocol is dominated by $(\log^2 n, n \log n)$ in the worst case. The verifier update cost on each step is bounded as $O(n^2)$.

5.8.5 Testing Bipartiteness

As a corollary of Theorem 5.6 it is also possible to test whether a graph is bipartite. This follows by applying the connectivity verification protocol described before on the both input graph G and the *bipartite double cover* of G, say G'. The bipartite double cover of a graph is formed by making two copies u_1, u_2 of every node u of G and adding edges $\{u_1, v_2\}$ and $\{u_2, v_1\}$ for every edge $\{u, v\}$ of G. It can be easily shown that G is bipartite if and only if the number of connected components in the double cover G' is exactly twice the number of connected components in G.

Corollary 5.2. *Given an input graph G with n vertices, there exists a SIP protocol for testing bipartiteness on G with* $(\log n)$ *rounds of communication, and* $(\log^2 n, n \log n)$ *cost.*

We note here that while we could have used known parallel algorithms for connectivity and MST combined with the protocol of Goldwasser et al. [89] and the technique of Cormode, Thaler and Yi [66] to obtain similar results, we need an explicit and simpler protocol with an output that we can fit into the overall TSP protocol described later in next section.

5.9 Streaming Interactive Proofs for Approximate Metric TSP

We can apply our protocols to another interesting graph streaming problem: that of computing an approximation to the min cost travelling salesman tour. The input here is a weighted complete graph of distances.

We briefly recall the Christofides heuristic: compute a MST T on the graph and add to T all edges of a min-weight perfect matching on the odd-degree vertices of T. The classical

Christofides result shows that the sum of the costs of this MST and induced min-weight matching is a $3/2$ approximation to the TSP cost. In the SIP setting, we have protocols for both of these problems. The difficulty however is in the dependency: the matching is built on the odd-degree vertices of the MST, and this would seem to require the verifier to maintain many more states as in the streaming setting. We show that this is not the case, and in fact we can obtain an efficient SIP for verifying a $(3/2 + \epsilon)$-approximation to the TSP.

Assume T is a $(1 + \varepsilon)$ approximate MST provided by the prover in the verification protocol and let T^* be the optimum MST on G. Also, let A be the optimum solution to TSP. Since graph G is connected, we have $w(A) \geq w(T^*)$ and because $(1 + \epsilon) \cdot w(T^*) \geq w(T)$, thus $(1 + \epsilon) \cdot w(A) \geq w(T)$. Further, let M be the min-cost-matching over the odd degree set O. By a simple reasoning, we can show that $w(M) \leq \frac{w(A)}{2}$, thus $w(M) + w(T) \leq (1 + \epsilon) \cdot w(A) + \frac{w(A)}{2}$ and from the triangle inequality it follows that the algorithm can verify the TSP cost within $(\frac{3}{2} + \epsilon)$-approximation.

We use first the protocol for verifying approximate MST described in Section 5.8. What remains is how we verify a min-cost perfect matching on the odd-degree nodes of the spanning tree. We employ the procedure described in Section 5.7 for maximum weight matching along with a standard equivalence to min-cost perfect matching. In addition to validating all the LP constraints, we also have to make sure that they pertain solely to vertices in ODD. We do this as above by using the fingerprint for ODD to ensure that we only count satisfied constraints on edges in ODD.

Here we describe the protocol for verifying approximate metric TSP in full details, which results in Theorem 5.7:

5.9.1 Protocol: TSP Verification

1. P presents a spanning tree which is claimed to be MST and can be verified within $(1 + \epsilon)$-approximation by V (as described in Section 5.8). V maintains a fingerprint on the vertices by using the MSE algorithm and updating the frequency of each vertex seen as an endpoint of an edge in the tree. This results in a fingerprint where each vertex has multiplicity equal to its degree in the MST.
2. P then lists all vertices of the spanning tree in lexicographic order annotated with their

degree. V verifies that this fingerprint matches the one constructed in the previous step and builds a new fingerprint for the set ODD of all odd-degree vertices (disregarding their degree).

3. P presents a claimed min-weight perfect matching on the vertex set ODD
4. V verifies that this list of edges is indeed a matching using the protocol from Section 5.5. In addition, it verifies that the vertices touched by these edges comprise ODD by using MSE to validate the fingerprint from the previous step.
5. To verify the lower bound on min-weight perfect matching, we first reduce to max-weight matching. Let $W = n^c$ be the a priori upper bound on the weight of each edge. Replace all weights w by $W + 1 - w$. Clearly now on a complete graph the max-weight matching corresponds to the min-weight perfect matching.
6. First, V needs to ensure that the entire certificate C is contained inside the ODD set. Recall that V has maintained a fingerprint of ODD, so we may use a variant of MSE. P replays C, along with any vertices which are in ODD but not in C and V checks the fingerprints match.
7. Then, V needs to check that all the constraints for the problem are satisfied by the certificate. This step is identical to what we described before for Maximum-Weight-Matching 5.7 (counting the "good" tuples), but here the satisfied constraints must be counted only on ODD set.
8. For this goal, we amend the protocol of Section 5.7 so that P streams the subset $V - ODD$ to the verifier and then V can simply decrement the frequency of all the tuples defined on $V - ODD$ by 1. Now all tuples corresponding to edges not containing both endpoints in ODD may achieve frequency at most 4 and hence will not be counted by the F_5^{-1} query.
9. Again, the accuracy of the claimed $V - ODD$ can be checked by using MSE. Let D be the claimed $V - ODD$. P streams D to V, which checks by MSE that the fingerprint of $D \cup ODD$ matches that of the entire vertex set. (Note that fingerprints are linear, so the fingerprint of $D \cup ODD$ is just the fingerprint of D plus the fingerprint of ODD.)
10. Now, V accepts the max-weight matching certificate if and only if the number of "good" tuples (which determines the count of satisfied edge constraints) is $\binom{|ODD|}{2}$ (i.e., the number of edges in complete graph induced by the ODD set). As discussed

earlier, these correspond to the value of F_5^{-1} in our extended universe.

Finally, the approximate TSP cost is the sum of the min-weight perfect matching on ODD and the MST cost on the graph.

Theorem 5.7. *Given a weighted complete graph with n vertices, in which the edge weights satisfy the triangle inequality, there exists a streaming interactive protocol for verifying optimal TSP cost within ($\frac{3}{2} + \epsilon$)-approximation with $(\log n)$ rounds of communication, and $(\log^2 n, n \log^2 n/\varepsilon)$ cost.*

5.10 Boolean Hidden Hypermatching and Disjointness

Boolean Hidden Matching (BHH_n^t) is a two-party one-way communication problem in which Alice's input is a boolean vector $x \in \{0,1\}^n$ where $n = kt$ for some integer k and Bob's input is a (perfect) hypermatching M on the set of coordinates $[n]$, where each edge M_r contains t vertices represented by indices as $\{M_{r,1}, ..., M_{r,t}\}$, and a boolean vector w of length $\frac{n}{t}$. We identify the matching M with its edge incidence matrix. Let Mx denote the length $\frac{n}{t}$ boolean vector $(\bigoplus_{1\leq i\leq t} x_{M_{1,i}}, \cdots, \bigoplus_{1\leq i\leq t} x_{M_{\frac{n}{t},i}})$. It is promised in advance that there are only two separate cases:

YES case: The vector w satisfies $Mx \bigoplus w = 0^{\frac{n}{t}}$.

NO case: The vector w satisfies $Mx \bigoplus w = 1^{\frac{n}{t}}$.

The goal for Bob is to differentiate these two cases.

The following lower bound result for BHH_n^t is obtained in [154]:

Lemma 5.3. *([154]) Any randomized one-way communication protocol for solving BHH_n^t when $n = kt$ for some integer k, with error probability at most $\frac{1}{4}$ requires $\Omega(n^{1-\frac{1}{t}})$ communication.*

Lemma 5.4. *Consider the streaming version of BHH_n^t problem, in which the binary vector x comes in streaming, followed by edges in M along with the boolean vector w for weights. There exists a streaming interactive protocol with communication and space cost $O(t \cdot \log n(\log\log n))$ for BHH_n^t problem.*

Proof. Considering the promise that we have in YES and NO case of BHH_n^t communication problem, it is enough to query the weights of vertices on only one of the hyperedges on the matching and compare it to the corresponding weight in vector w. This way the BHH_n^t problem can be reduced to t instances of INDEX problem. Assume the vector x as the input stream and take one of the followed hyperedges, say $M_r = \{M_{r,1}, ..., M_{r,t}\}$, as the t query index. In this scenario the verifier just need to apply the verification protocol INDEX in t locations $\{M_{r,1}, ..., M_{r,t}\}$ on x and check if $\bigoplus_{1 \leq i \leq t} x_{M_{r,i}} \bigoplus w_r = 0$ or $\bigoplus_{1 \leq i \leq t} x_{M_{r,i}} \bigoplus w_r = 1$. According to [49], the verification (communication and space) cost for INDEX problem is $O(\log n(\log \log n))$ and this results in $O(t \cdot \log n(\log \log n))$ cost for BHH_n^t. □

We now show a similar result for Disjointness($DISJ_n$). $DISJ_n$ is a two-party one-way communication problem in which Alice and Bob each have a boolean vector x and $y \in \{0, 1\}^n$, respectively, and they wish to determine if there is some index i such that $a_i = b_i = 1$. Razoborov [141] shows an $\Omega(n)$ lower bound on the communication complexity of this problem for one-way protocols. We show now however that $DISJ_n$ is easy in the SIP model.

Lemma 5.5. *Consider the streaming version of $DISJ_n$ problem, in which the binary vector x comes in streaming, followed by binary vector y. There exists a streaming interactive protocol with communication and space cost $O(\log^2 n)$ for $DISJ_n$.*

Proof. The verifier maintains a universe $\mathcal{U}$ corresponding to $[n]$. When the verifier sees the i^{th} bit of x, it increments the frequency of universe element i by x_i. Now when the verifier streams y_i, the verifier again increments the frequency of element i by y_i. Clearly, x and y correspond to disjoint sets if and only if $F_2^{-1} = 0$. We can then simply run the Finv protocol, and the bound follows by Lemma 4.1 □

Lemma 5.4 and 5.5 shows that while BHH_n^t and $DISJ_n$ are lower bound barriers to computations in the streaming model, however they are easily tractable in the streaming verification setting. This gives a first suggestion that for problems such as MAX-CUT and MAX-MATCHING where most of the known lower bounds go through BHH_n^t or $DISJ_n$, streaming verification protocols may prove more effective, as was the initial motivation for our study.

5.11 Verifier Update Time Complexity

Note that while all the graph verification protocols presented in this chapter achieve very small space and communication costs, the update time could be high (polynomial in n) since processing a single stream token may trigger updates in many entries of $\mathbf{a}$. But by using a nice trick found in [49], the verifier time can be reduced to polylog n. Here we state the main results which can be applied to all the verification protocols in this chapter to guarantee polylog n verifier update time.

Lemma 5.6. *Assume a data stream τ in which each element triggers updates on multiple entries of vector* $\mathbf{a}$*, and each entry in this vector is indexed by a multidimensional vector with b coordinates and let* $\mathcal{U} \subseteq [n^c]^b$*. In all the SIP protocols for graph problems in this chapter, the updates in the form of* $\mathbf{a}[(\beta_1, \beta_2, \cdots, \beta_q, *, \cdots, *)]$ *(which is interpreted as: update all entries β where* $\beta \in \{(\beta_1, \cdots, \beta_q, s_1, \cdots, s_{b-q}) | s_i \in [n^c], i \in [b-q], (\beta_1, \cdots, \beta_q, s_1, \cdots, s_{b-q}) \in \mathcal{U}\}$*) can be done in poly*log n *time.*

Here we present the proof for Lemma 5.6. The main ideas are extracted from [49], in which this trick is used for reducing verifier time in Nearest Neighbor verification problem. For more details, refer to Section 3.2 in [49].

Proof. Suppose the boolean function ϕ which takes two vectors $\boldsymbol{\beta}$ and $\mathbf{x}$ as inputs, in which $\boldsymbol{\beta} = (\beta_1, \cdots, \beta_b)$ is a vector with b coordinates each $\beta_i \in [n]^c$ and $\mathbf{x} = (x_1, \cdots, x_q)$ is a vector with $q < b$ coordinates each $x_i \in [n]^c$. Here we assume $\boldsymbol{\beta}$ is an index in the vector $\mathbf{a}$ defined over the input stream and $\mathbf{x}$ the update vector defined by the current stream element(i.e. specifies which indices in $\mathbf{a}$ must be updated). Define $\phi(\boldsymbol{\beta}, \mathbf{x}) = 1 \leftrightarrow \beta_i = x_i, 1 \leq \forall i \leq q$ with $O(\log n)$-bits inputs (since we can assume b as a small constant). Let define the length of the shortest de Morgan formula for function ϕ as $\text{fsize}(\phi)$. Obviously, the function ϕ is essentially the equality check on O(logn)-bits input and we know that the addition and multiplication of s-bits inputs can be computed by Boolean circuits in depth $\log s$, resulting in Boolean formula of size $\text{poly}(s)$. Thus, $\text{fsize}(\phi) = \text{polylog}\, n$. Considering the boolean formula for ϕ, we associate a polynomial $\tilde{G}$ with each gate G of this formula, with input variables $W_1, \cdots, W_{b \log n}$ and $X_1, \cdots, X_{q \log n}$, as follows:

$$G = \beta_i \Rightarrow \tilde{G} = W_i$$

$$G = x_i \Rightarrow \tilde{G} = X_i$$

$$G = \neg G_1 \Rightarrow \tilde{G} = -\tilde{G_1}$$

$$G = G_1 \wedge G_2 \Rightarrow \tilde{G} = \tilde{G_1}\tilde{G_2}$$

$$G = G_1 \vee G_2 \Rightarrow \tilde{G} = 1 - (1 - \tilde{G_1}(1 - \tilde{G_2}))$$

Let $\tilde{\phi}(W_1, \cdots, W_{b\log n}, X_1, \cdots, X_{q\log n})$ to be the polynomial associated with the output gate, which is in fact the standard arithmetization of the formula. We consider $\tilde{\phi}$ as a polynomial defined over $\mathbb{F}[W_1, \cdots, W_{b\log n}, X_1, \cdots, X_{q\log n}]$ for a large enough finite field $\mathbb{F}$. By construction, $\tilde{\phi}$ has total degree at most fsize(ϕ) and agree with ϕ on every Boolean input. Define the polynomial $\Psi(W_1, \cdots, W_{b\log n}) = \Sigma_{i=1}\tilde{\phi}((W_1, \cdots, W_{b\log n}), \mathbf{x}^{(i)})$, in which $\mathbf{x}^{(i)}$ is the update vector defined by the element i in the stream. Now we can observe that the vector **a** defined by the stream updates, can be interpreted as follows:

$$a[\boldsymbol{\beta}] = \Sigma_{i=1}\phi(\boldsymbol{\beta}, \mathbf{x}^{(i)}) = \Sigma_{i=1}\tilde{\phi}(\boldsymbol{\beta}, \mathbf{x}^{(i)}) = \Psi(\boldsymbol{\beta})$$

It follows that Ψ is the extension of **a** to $\mathbb{F}$ with degree equal to fsize(ϕ) and can be defined implicitly by input stream. Also, the verifier can easily evaluate $\Psi(\mathbf{r})$ for some random point $\mathbf{r} \in \mathbb{F}^{b\log n}$, as similar to polynomial evaluation in Sum-Check protocol. Considering that fsize(ϕ) = polylog n, the complexity result of update time follows. Note that this approach adds an extra space cost fsize(ϕ) =poly log n for the size of Boolean formula, but in general this does not affect the total space cost of the protocols discussed in this chapter. □

5.12 Revisit the Graph Protocols with Constant-Rounds Communication

Note that sum-check is used as the core of all the graph verification protocols. In this section, we study the possibility of reducing the round-complexity of these protocols to constant-rounds using the results stated in Section 4.3. In all the $(\log n)$-rounds verification protocols which we presented before the space cost is $\log^2 n$ bits and with changing the protocol to constant γ-rounds, we improve the space to $O(\log n)$ bits. On the other hand, in most of these protocols the communication cost is dominated by the size of *certificate*, which is generally bounded by $O(n \log n)$. Thus, while using constant-round sum-check as the core of verification protocols will increase the related communication cost by a $n^{\frac{1}{\gamma}}$ factor, but that will not change the total communication cost of SIPs for matching and TSP, in which the communication cost is dominated by the certificate size.

CHAPTER 6

STREAMING VERIFICATION ALGORITHMS FOR DATA ANALYSIS

In this chapter, we initiate a study of streaming interactive protocols for problems in data analysis.

6.1 Technical Overview

The main components of our technical contribution are as follows.

(i) We give an improved result for rectangular matrix multiplication. We use this in a new annotated data streaming protocol for verifying *eigenvectors* of a symmetric matrix presented as a stream of updates to entries. Verifying the eigenstructure of a matrix is an important subroutine in many matrix analysis problems, and this result is likely to be of independent interest (Section 6.3). Concretely, our protocol can verify k eigenvectors of an $n \times n$ matrix, with communication cost $k^2 \cdot h$ and space cost v, for any desired pair of positive integers h, v satisfying $h \cdot v \geq n$. This improves on the communication cost of prior work by a factor of k. In our annotated data streaming protocol for matrix multiplication, we first observe that multiplying a $k \times n$ matrix A with an $n \times k'$ matrix B is equivalent to performing k' *matrix-vector* multiplications, one for each column of B. One could naively verify these products by simply executing k' independent instances of the matrix-vector verification protocol from prior work [64]. We show how to improve on this naive solution by exploiting the fact that the k' matrix-vector multiplications are not independent, because the matrix A is held fixed in all of them. This leads to an improved subroutine for rectangular matrix multiplication that in turn allows us to verify eigenvectors of a matrix.

(ii) We give *3-message* SIPs that can verify a minimum enclosing ball and the width of a point set *exactly* with polylogarithmic space and communication costs. Note that it is known that the MEB cannot be approximated to better than a $\sqrt{2}$ factor by a streaming algorithm with polylogarithmic space: this provides an example where a SIP is strictly exponentially more efficient than any streaming algorithm (other examples were known previously). Separately, we show that the hardness result for the MEB problem holds even when the points are chosen from a discrete cube: this is important because our interactive proofs require discrete input (Section 6.4). (iii) We also show a simple 3-message protocol for verifying a 2-approximation to the k-center in a metric space, via simple adaptation of the Gonzalez 2-approximation for k-center (Section 6.4). (iv) We present polylogarithmic round protocols with polylogarithmic communication and verifier space for verifying optimal k-centers and k-slabs in Euclidean space (note that computing the MEB and width of a point set correspond to the 1-center and 1-slab problems, respectively). In order to do this, we use a *prefix* sum-check protocol that might be of independent interest (Section 6.5). For the k-center and k-slab problems, our verification protocols consist of checking that the claimed solution is both feasible and optimal. We show how to verify feasibility by reducing to a carefully constructed instance of the Range Counting problem; we then apply a 2-message SIP for Range Counting due to prior work [48]. Optimality, on the other hand, is harder to verify, because the prover must convince the verifier that *no* other feasible solution has lower cost. When $k = 1$, we show that there is a sparse *witness of optimality*, which the verifier can check directly using 3 messages, by reduction to Range Counting. For general k, we cannot show that there is a sparse witness of optimality. However, we observe that the “for-all” constraint on feasible solutions can be expressed as a sum over all solutions of lower cost. Choosing a cost-based ordering of solutions converts this into a partial sum over a prefix of the ordered set of solutions. Our main tool is a way to verify such a sum in general, using polylogarithmically many messages, *even when the relevant prefix is only known after the stream has passed*.

6.2 Preliminaries

The streaming verification protocol proposed in this chapter also follow the two models SIPs and annotated data streams which were introduced in Chapter 4 and we skip the details here.

6.2.1 Input Model

All of the protocols we consider can handle inputs specified in a general data stream form. Each element of the stream is a tuple (i, δ), where each i lies in a data universe $\mathcal{U}$ of size u, and $\delta \in \{+1, -1\}$. Negative values of δ model deletions. The data stream implicitly defines a frequency vector $\mathbf{a} = (a_1, \ldots, a_u)$, where a_i is the sum of all δ values associated with i in the stream.

6.2.2 Protocol Costs

As stated in Chapter 4, there are two principal costs associated with a SIP: the space used by with the verifier, and the total amount of communication, expressed as the number of bits exchanged between V and P. Our goal will be to ensure that V uses sublinear space and that the protocol uses sublinear communication, because in settings involving massive data, it is essential that V and P avoiding shipping around the entire input. Of course, we will also desire protocols in which V and P can run quickly. In most of our protocols, both V and P can run in time quasilinear in the size of the input stream.

6.2.3 Discretization

The protocols we employ make extensive use of finite field arithmetic. In order to apply these techniques to geometric problems, we must assume that all input points are drawn from the $[0, 1]^d$ cube, discretized to form a grid $[m]^d$. That is, we assume that the data universe for these problems is $\mathcal{U} = [m]^d$. Thus all points have coordinates of the form $j/m, j \in \mathbb{Z}$. It also follows that the distance $D(\mathbf{x}, \mathbf{y})$, between any two points $\mathbf{x}, \mathbf{y} \in [m]^d$ is a multiple of $\varepsilon \geq 1/m^d$. Importantly, the costs of our protocols will depend only logarithmically on m, enabling the grid to be exceedingly fine while still yielding tractable costs.

6.3 Verifying Matrix Eigenstructure

Many algorithms in data analysis (principal component analysis and multidimensional scaling, to mention two of the most prominent) require computation of the *eigenpairs* (eigenvalues and eigenvectors) of a large data matrix. Eigenvalues of a streamed $n \times n$ matrix can be computed approximately without a prover [19], but there are no streaming algorithms to compute the *eigenvectors* of a matrix, mainly because storing these can be costly.

Verifying the eigenstructure of a symmetric matrix A is more difficult than merely verifying that a claimed $(\lambda, \mathbf{v})$ is an eigenpair. This is because the prover must convince the verifier not only that each $(\lambda_i, \mathbf{v}_i)$ satisfies $A\mathbf{v} = \lambda\mathbf{v}$, but that the collection of eigenvectors together are orthogonal. For example, the prover could supply repeated copies of the same eigenvectors. Or if there is a repeated eigenvalue, then it could generate repeated vectors in the linear span of the eigenvectors of this eigenvalue and claim them as independent eigenvectors.

The key here is *orthogonality*: the prover must prove that $VV^\top = D$ where V is the collection of eigenvectors and D is some diagonal matrix. Note however that this matrix multiplication check is *rectangular*: if we wish to verify that a collection of k eigenvectors are orthogonal, we must multiply a $k \times n$ matrix V by an $n \times k$ matrix $V^\top$, the result being a $k \times k$ matrix. We present an annotation protocol called MatrixMultiplication to verify such a *rectangular* matrix multiplication.

Theorem 6.1. *Let A be a $k \times n$ matrix and B an $n \times k'$ matrix, both with entries in a finite field $\mathbb{F}$ of size $6n^3 \leq |\mathbb{F}| \leq 6n^4$. Let (h, v) be any pair of positive integers such that $h \cdot v \geq n$. There is a annotated data streaming protocol for computing the product matrix $C = A \cdot B$ with communication cost $O(k \cdot k' \cdot h \cdot \log n)$ bits and space cost $O(v \cdot \log n)$ bits.*

In particular, by setting $h = \max\left(1, \sqrt{n/(k \cdot k')}\right)$, and $v = \min\left(n, \sqrt{k \cdot k' \cdot n}\right)$ in Theorem 6.1, one obtains a protocol in which the communication cost is as follows:

$$O\left(\max\left(k \cdot k', \sqrt{k \cdot k' \cdot n}\right) \cdot \log n\right)$$

and the space cost is $O\left(\min\left(n, \sqrt{k \cdot k' \cdot n}\right) \cdot \log n\right)$. With this setting of parameters, *both* the communication and space costs are sublinear in the input size $(k + k') \cdot n \cdot \log n$ if *either* k or k' is in $o(n)$. In contrast, when both k, k' are in $\Omega(n)$, the amount of communication required just to specify the answer C is linear in the input size. Thus, whenever it is even conceivable to have communication and space costs both be sublinear in the input size, our matrix multiplication protocol achieves it. Most importantly, Theorem 6.1 is strictly better than doing repeated matrix-vector verifications [64] by a factor of k in the communication cost.

Before proving Theorem 6.1, we first show how to use Theorem 6.1 to verify that a claimed set of k eigenvalues and eigenvectors are indeed eigenpairs of a given symmetric matrix A.

Problem 6.2 (Verifying eigenpairs)**.** *Let A be a real symmetric matrix that is updated by streaming entries of the form (i, j, δ_{ij}) which encode the update $A_{ij} = A_{ij} + \delta_{ij}$. The prover returns a set of k pairs $(\lambda_i, \mathbf{v}_i)$ and claims that these are eigenpairs of A.*

6.3.1 The Eigenpair Verification Protocol

The eigenpair verification protocol invokes MatrixMultiplication twice. In the first invocation, MatrixMultiplication is used to simultaneously verify that all claimed eigenpairs are indeed eigenpairs (i.e., to check that $A \cdot V = D \cdot V$, where V is the matrix whose ith column equals $\mathbf{v}_i$, and D is some diagonal matrix). In the second invocation, MatrixMultiplication is used to check that the claimed eigenvectors are orthogonal, by verifying that $V^\top V = D'$ for some diagonal matrix D' provided by the prover.

Theorem 6.3. *Let A be a symmetric $n \times n$ matrix over a field $\mathbb{F}$ of size $6n^3 \leq |\mathbb{F}| \leq 6n^4$, let $k \leq rank(A)$ be an integer, and let h, v be positive integers satisfying $h \cdot v \geq n$. Then there is an annotated data streaming protocol for verifying a collection of k eigenpairs that has total communication cost $O(k^2 \cdot h \log n)$ and requires verifier space $O(v \cdot \log n)$, where $\alpha > 0$.*

Just as with Theorem 6.1 itself, if $k = o(n)$, then it is possible to choose h, v in Theorem 6.3 to ensure that both the communication and space costs are sublinear in the input size.

Proof of Theorem 6.1. Our protocol builds on the optimal annotations protocols for inner product and matrix-vector multiplication from [47] and [64]. In order to compute the inner product between two vectors $a, b \in \mathbb{R}^n$, the verifier treats the n entries of a and b as a grid $[h] \times [v]$, and considers the unique bivariate polynomials $\widetilde{a}(X, Y)$ and $\widetilde{b}(X, Y)$ over $\mathbb{F}$ of degree at most h in X and v in Y satisfying $\widetilde{a}(x, y) = a(x, y)$ and $\widetilde{b}(x, y) = b(x, y)$ for all $(x, y) \in [h] \times [v]$. The verifier picks a random $r \in \mathbb{F}$, and evaluates $\tilde{a}(r, y)$ and $\tilde{b}(r, y)$ for all $y \in [v]$. As observed in [47], the verifier can compute $\tilde{a}(r, y)$ for any $y \in [v]$ in space $O(\log |\mathbb{F}|)$, with a single streaming pass over the input. Hence, the verifier's total space usage is $O(v \cdot \log |\mathbb{F}|)$. The prover then sends a univariate polynomial $s(X)$ of degree at most h, claimed to equal $g(X) = \sum_{y \in [v]} \tilde{a}(X, y) \cdot \tilde{b}(X, y)$. The verifier accepts $\sum_{x \in [h]} s(X)$ as the correct answer if and only if $s(r) = \sum_{y\in[v]} \tilde{a}(r,y) \cdot \tilde{b}(r,y)$.

Let us denote the rows of A by $\mathbf{a}_1, \ldots, \mathbf{a}_k$ and the columns of B by $\mathbf{b}_1, \ldots, \mathbf{b}_k$. Notice that each entry C_{ij} of C is the inner product of $\mathbf{a}_i$ and $\mathbf{b}_j$.

6.3.2 The Prover's Computation

In our matrix multiplication protocol, the prover simply runs the above inner product protocol $k \cdot k'$ times, one for each entry C_{ij} of C. This requires sending $k \cdot k'$ polynomials, $s_{ij}(X)\colon (i,j) \in [k] \times [k']$, each of degree at most h. Hence, the total communication cost is $O(k \cdot k' \cdot h \cdot \log n)$.

6.3.3 The Verifier's Computation while Observing Entries of A

The verifier does not keep separate state for each row of A as she would if she were to running a separate inner product protocol for each row of A. Rather the verifier picks a random α in the relevant finite field, for each $y \in [v]$, the verifier computes a *fingerprint* of the values $\{\tilde{a}_i(r,y)\}$, as i ranges from 1 to k, using the random value α to define the fingerprinting function. That is, the verifier picks a random α and computes, for each $y \in [v]$, the quantity $s_y := \sum_i \tilde{a}_i(r,y)\alpha^i$. Using standard techniques [47], the verifier can compute each s_y with a single streaming pass over the entries of A, in $O(\log n)$ space. Hence, the verifier can compute all of the s_y values in total space $O(v \log n)$.

6.3.4 The Verifier's Computation while Observing Entries of B

As the verifier observes the vectors $b_1, \ldots, b_{k'}$, she computes a fingerprint for each $y \in [v]$ of the $\tilde{b}(r,y)$ values. But in this fingerprint she replaces the random value α with α^k. More specifically, for each $y \in [v]$, the verifier computes the quantity $s'_y := \sum_{j\in k'} \tilde{b}_j(r,y)\alpha^{k\cdot j}$. The reason that we define s'_y in this way is because it ensures that $s_y \cdot s'_y = \sum_{(i,j)\in[k]\times[k']} \tilde{a}_i(r,y) \cdot \tilde{b}_j(r,y)\alpha^{k\cdot j+i}$, which is just a fingerprint of the set of values $\{\tilde{a}_i(r,y) \cdot \tilde{b}_j(r,y)\}$ as (i,j) ranges over $[k] \times [k']$.

To check that all s_{ij} polynomials are as claimed, the verifier does the following. As the verifier reads the s_{ij} polynomials, she computes a fingerprint of the $s_{i,j}(r)$ values, i.e., the verifier computes $\sum_{i,j} s_{i,j}(r) \cdot \alpha^{j\cdot k+i}$. The verifier checks whether this equals $\sum_y (s_y \cdot s'_y)$. If so, the verifier is convinced that $A_{ij} = \sum_{x\in[h]} s_{ij}(x)$ for all $(i,j) \in [k] \times [k']$. If not, the verifier rejects. The protocol is complete and has is sound with constant error:

6.3.5 Proof of Completeness

If the $s_{i,j}$ polynomials are as claimed, then:

$$\begin{aligned}\sum_{i,j\in[k]\times[k']} g_{i,j}(r)\cdot\alpha^{j\cdot k+i} &= \sum_{i,j\in[k]\times[k']}\sum_{y\in[v]} \tilde{a}_i(r,y)\cdot\tilde{b}_j(r,y)\alpha^{j\cdot k+i}\\ &= \sum_{y\in[v]}\sum_{i,j\in[k]\times[k']} \tilde{a}_i(r,y)\cdot\tilde{b}_j(r,y)\alpha^{j\cdot k+i} = \sum_{y\in[v]} s_y\cdot s'_y.\end{aligned}$$

6.3.6 Proof of Soundness

If any of the $s_{i,j}$ polynomials are *not* as claimed (i.e., if $s_{ij}(X) \neq g_{ij}(X)$ as formal polynomials), then with probability at least $1-h/|\mathbb{F}|$ over the random choice of $r\in\mathbb{F}$, it will hold that $s_{i,j}(r)\neq g_{ij}(r)$. In this event the verifier will wind up comparing the fingerprints of two different vectors, namely the $k\cdot k'$-dimensional vector whose (i,j)'th entry is $s_{i,j}(r)$, and the $k\cdot k'$-dimensional vector vector whose (i,j)'th entry is $\sum_{y\in[v]}\tilde{a}_i(r,y)*\tilde{b}_j(r,y)$. These fingerprints will disagree with probability at least $1-k\cdot k'/|\mathbb{F}|$. Hence, the total probability that the prover convinces the verifier to accept is at most $h/|\mathbb{F}|+k\cdot k'/|\mathbb{F}|$. If $|\mathbb{F}|\geq 100\cdot h\cdot k\cdot k'$, the soundness error will be bounded by $1/50$. □

6.3.7 On V's and P's Runtimes

Using Fast Fourier Transform techniques (cf. [63, Section 2]), the prover the protocol of Theorem 6.1 can be made to run in $O(k\cdot k'\cdot n\log n)$ total time, assuming the total number of updates to the input matrices A, B is $O(k\cdot k'\cdot n\log n)$. The verifier can be made to run in time $O(\log n)$ per stream update.

6.4 Solving Geometric Problems in a Few Rounds

In this section, we give 3-message SIPs of polylogarithmic cost for the problems of finding a Minimum Enclosing Ball, and for computing the width of a point set. The key to obtaining constant-round protocols for these problems lies in identifying a a sparse dual witness that proves optimality (or near-optimality) of the claimed (primal) solution. The verifier is then convinced of both the feasibility and optimality of the claimed primal solution, so long as V can confirm that the primal and dual solutions are both feasible. We show how the verifier can perform both feasibility checks via a careful reduction to an instance of the RangeCount problem (for which an $O(1)$-round SIP of logarithmic cost was given in prior work; cf. Theorem 4.3).

6.4.1 Minimum Enclosing Balls

Consider the Euclidean k-center problem with $k = 1$, otherwise known as the MEB: given a set of n points P in $\mathbb{R}^d$ and distance function D, find a ball B^* of minimum radius that encloses all of them. As stated in Section 6.2, we assume a grid structure $\mathcal{U} = [m]^d$ over the space of the possible points, where d is the dimensionality of the data and for all $\mathbf{x}, \mathbf{y} \in [m]^d, D(\mathbf{x}, \mathbf{y}) \leq 1$ is an integer multiple of a small parameter $\epsilon \geq \frac{1}{m^d}$.

The MEB presents an interesting contrast between our model and the classical streaming model. It is known that no streaming algorithm that uses polynomial (in d) space can approximate the MEB of a set of points to better than a factor of $\sqrt{2}$ [7]. However, we will show here that the MEB can be computed to within an ϵ *additive* error by a 3-message SIP using space and communication that grow just logarithmically in m and $1/\varepsilon$ (and linearly in the dimension d of the input point set). In contrast, the best streaming *multiplicative* $(1+\epsilon)-$approximation for the MEB uses $(1/\epsilon)^d$ space.

6.4.2 Protocol

At the start of the protocol, the prover first sends the (claimed) minimum enclosing ball B for the input point set. Our protocols reduces checking feasibility and optimality of B to carefully constructed instances of the RangeCount problem.

6.4.3 Checking Feasibility

We consider a new range space, in which the range set $\mathcal{B}$ is defined to consist of all balls with radius $j\epsilon$: $j \in \{0, 1, \ldots, m^d\}$ and with centers in $[m]^d$. Notice that $|\mathcal{B}| = O(m^{2d})$. Using the protocol for RangeCount (Theorem 4.3), we can verify that the claimed solution B does in fact cover all points (because this will hold if and only if the range count of B equals the cardinality of the input point set $|P| = n$).

6.4.4 Checking Optimality

We will make use of the following well known fact about minimal enclosing balls first proved by Goel, Indyk and Varadarajan [73, 84]:

Lemma 6.1. *Let B^* be the minimal enclosing ball of a set of points P in $\mathbb{R}^d$. Then there exist at most $d+2$ points of P that lie on the boundary ∂B^* of B^* and contain the center of B^* in their* *convex hull.*

Sketch. This certificate of optimality can be constructed by writing the linear program for the MEB and considering its dual. Complementary slackness gives us the desired sparse witness. □

6.4.5 Putting it All Together

The complete 3-message MEB protocol works as follows.

1. V processes the data stream for RangeCount (with respect to $\mathcal{B}$ and P).
2. P computes the MEB B^* of P, then rounds the center c of the MEB to the nearest grid vertex. Denote this vertex by c^*. P sends c^* to V, as well as the radius r of B^*, and a subset of points $T \in P$ in which $\text{MEB}(T) = \text{MEB}(P)$. (Note that based on Lemma 6.1, $|T| \leq d + 2$ suffices).
3. V first computes the center c of the MEB for the subset T and checks if c^* is actually the rounded value of c. Then V runs a RangeCount protocol with P to verify that the ball of radius $r + \epsilon$ and center c^* contains all of the input points. It then runs multiple copies of PointQuery to verify that the subset $|T| \leq d + 2$ points provided by P are actually in the input set P.

Theorem 6.4. *There exists a 3-message SIP for the Minimum Enclosing Ball (MEB) problem with communication and space cost bounded by* $O(d^2 \log^2 m)$.

6.4.6 On V's and P's Runtimes

Assuming the distance function D under which the instance of MEB is defined satisfies mild "efficient-computability" properties, both V can be made to run in total time $\text{polylog}(m^d)$ per stream update in the protocol of Theorem 6.4. Specifically, it is enough that for any point $\mathbf{x} \in P$, there is a De-Morgan formula of size $\text{polylog}(m^d)$ that takes as input the binary representation of a ball $B \in \mathcal{B}$ and outputs 1 if and only if $\mathbf{x} \in B$. Under the same assumption on D, the prover P can be made to run in time $T + n \cdot \text{polylog}(m^d)$, where T is the time required to find the MEB of the input point set P. For details, we direct the interested reader to the full description of the PointQuery protocol of [48].

6.4.7 Streaming Lower Bounds on the Grid

A priori, it may be possible that our assumption that the input points all lie on a grid $[m]^d$ makes the MEB problem easy, in the sense that it may be solvable by a small-space streaming without access to an untrusted prover. Here we show that this is not the case, and that the standard lower bound for streaming MEB due to Agarwal and Sharathkumar [7] can be modified to work even if the points lie on a grid. The key lemma in Agarwal and Sharathkumar's lower bound is a construction of a collection of almost orthogonal vectors that are centrally symmetric. Let S^{d-1} denote the unit sphere in $\mathbb{R}^d$.

Lemma 6.2 (Agarwal and Sharathkumar [7]). *There is a centrally symmetric point set $K \subset S^{d-1}$ of size $\Omega(\exp(d^{\frac{1}{3}}))$ such that for any pair of distinct points $p, q \in K$ if $p \neq -q$, then*

$$\sqrt{2}(1 - \frac{2}{d^{\frac{1}{3}}}) \leq \|p - q\| \leq \sqrt{2}(1 + \frac{2}{d^{\frac{1}{3}}}) \tag{6.1}$$

This point set is then used by an adversary to "defeat" any algorithm claiming a $\sqrt{2} - \delta$ approximation. Note that the "almost orthogonal" property follows from the observation that for unit vectors p, q, $\|p - q\|^2 = 2 - 2\langle p, q\rangle$ and therefore the condition of the lemma above implies that $\langle p, q\rangle \leq \frac{4}{d^{\frac{1}{3}}}$

It turns out that this "almost-orthogonal" property can be achieved by vectors with integer coordinates. The proof is in the same spirit of the proofs that sign matrices can be used in the Johnson-Lindenstrauss lemma, and follows from an observation by Ryan O'Donnell [136]. We recreate the proof here for completeness.

Lemma 6.3 (Bernstein's inequality). *Let $X_1, \ldots, X_d$ be independent Bernoulli variables taking values in $\{+1, -1\}$ with equal probability. Then*

$$Pr[|\frac{1}{d}\sum_i X_i| \geq \epsilon] \leq 2\exp\left(-d\epsilon^2 / \left(2\left(1 + \epsilon/3\right)\right)\right).$$

Lemma 6.4. *Let $t = \exp(\frac{\epsilon^2 d}{4})$. Let $\mathbf{u}_1, \ldots, \mathbf{u}_t$ be random vectors in which each entry is set to $1/\sqrt{d}$ or $-1/\sqrt{d}$, with probability $\frac{1}{2}$ each. There is a positive probability of $|\langle \mathbf{u_i}, \mathbf{u_j}\rangle| \leq \epsilon$ holding for all $i \neq j$.*

Proof. We define variables x_{ij} as the Bernoulli variables corresponding to Lemma 6.4, where $i \leq k, j \leq d$. That is, define the x_{ij} variables such that:

$$\mathbf{u}_i = \frac{1}{\sqrt{d}}(x_{i1}, \ldots, x_{id}).$$

We want to analyze the behavior of $\langle \mathbf{u}_i, \mathbf{u}_j \rangle$. Set $Y_k^{ij} = x_{ik} x_{jk}$ and write $\langle \mathbf{u}_i, \mathbf{u}_j \rangle$ as $\frac{1}{d} \sum_k Y_k^{ij}$. Note that for each i, j, and k, Y_k^{ij} is a Bernoulli variable with range $\{-1, +1\}$, and for any fixed i, j, the variables Y_k^{ij} are independent. Therefore, we can apply Bernstein's inequality to the collection $\{Y_k^{ij}\}$ for a fixed i, j.

For simplicity, assume that $\epsilon \leq 1$. Then Bernstein's inequality implies that

$$\Pr[|\langle \mathbf{u}_i, \mathbf{u}_j \rangle| \geq \epsilon] \leq 2 \exp(-d\epsilon^2/4).$$

It follows that the probability that $|\langle \mathbf{u}_i, \mathbf{u}_j \rangle| \geq \epsilon$ is at most $2\exp(-d\epsilon^2/4)$. Now if we set $t = \exp(\frac{d\varepsilon^2}{4})$, then this probability value equals $\frac{2}{t^2}$ by choice of t and hence by taking a union bound over at most $\binom{t}{2} \leq \frac{t^2}{2}$ pairs of (i, j) we conclude that there is a positive probability of $|\langle \mathbf{u}_i, \mathbf{u}_j \rangle] \leq \epsilon$ holding for all $i \neq j$. □

6.4.8 Verifying the Width of a Point Set

The approach underlying the 3-message SIP for the MEB problem of Section 6.4.1 can be applied to another shape fitting problem that has been studied extensively in computational geometry. We define a *slab* as the region between a pair of parallel hyperplanes, with the *width* of the slab being the distance between the two hyperplanes (with respect to a fixed distance function D). The *width* of a point set is the minimum width of a slab that covers the entire set. Like the MEB problem, the width of a point set can be approximated by a streaming algorithm using $O(1/\epsilon^{O(d)})$ space [50], without access to a prove. Unlike the MEB, however, there are no known lower bounds for computing the width of a stream of points.

We describe an efficient constant-round SIP to *exactly compute* the width of a point set. As before, we study the problem in the discrete setting, i.e., we assume that the data stream elements are a subset of points over a grid structure $\mathcal{U} = [m]^d$. Let $\mathcal{R}$ denote the set of all the ranges defined by single slab (i.e., each range consists of the area between some two parallel hyperplanes).

6.4.9 Certificate of Optimality

Given a slab S that is claimed to be a minimal-width slab covering the input point set P, the following lemma (akin to Lemma 6.1) guarantees the existence of a sparse witness of optimality for S.

Lemma 6.5. *Given the input point P in d-dimension, every optimal-width single slab S consisting of the area between parallel hyperplanes h_1, h_2 covering P can be described by a set of $k + k' = d + 1$ points from input point set P, in which k points lie on the hyperplane h_1 and k' points lie on the hyperplane h_2.*

Proof. We express S as an optimal solution to a certain linear program. We then infer the existence of the claimed witness of optimality for S via strong linear programming duality and complementary slackness.

Assume the two hyperplanes specifying S are of the form $h_1 : \langle \mathbf{w}, \mathbf{x} \rangle = 1$ and $h_2 : \langle \mathbf{w}, \mathbf{x} \rangle = \ell$, where $\mathbf{w} \in \mathbb{R}^d$. Then the pair $(\mathbf{w}, \ell)$ corresponds to an optimal solution of the following linear program:

$$\begin{aligned} &\min \quad \ell \\ \text{s.t. } \forall i \in \{1, \ldots, |P|\} &\quad \langle \mathbf{w}, \mathbf{x}_i \rangle \geq 1 \\ \forall i \in \{1, \ldots, |P|\} &\quad \langle \mathbf{w}, \mathbf{x}_i \rangle \leq \ell \end{aligned}$$

We write the LP in the standard form:

$$\begin{aligned} &\max \quad -\ell \\ \text{s.t. } \forall i \in \{1, \ldots, |P|\} &\quad (-\mathbf{x}_i^T) \cdot \mathbf{w} \geq 1 \\ \forall i \in \{1, \ldots, |P|\} &\quad \mathbf{x}_i^T \cdot \mathbf{w} - \ell \leq 0 \end{aligned}$$

Let x_{ij} denote the jth entry of input point $\mathbf{x}_i \in [m]^d$. Standard manipulations reveal the dual.

$$\begin{aligned} &\min \quad \sum_{i=1}^{|P|} y_i \\ \text{s.t. } \forall j \in \{1, \ldots, d\} &\quad \sum_{i=1}^{|P|} (y_i - z_i) x_{ij} = 0. \\ &\quad \sum_{i=1}^{|P|} z_i = 1. \end{aligned}$$

Let $\mathbf{y} = (y_1, \ldots, y_{|P|})$ and $\mathbf{z} = (z_1, \ldots, z_{|P|})$ denote an optimal solution to the above dual.

Claim 6.5. *For any i, y_i and z_i cannot both be nonzero.*

Proof. By complementary slackness, y_i and z_i are both nonzero only if both of the corresponding primal inequalities are tight, which can only hold if the width is zero. □

Claim 6.6. *In total, the number of nonzero entries in y and z must be at least $d + 1$.*

Proof. Fix any $j \in \{1, \ldots, d\}$ and consider the constraint

$$\sum_{i=1}^{|P|} y_i x_{ij} = \sum_{i=1}^{|P|} z_i x_{ij} \tag{6.2}$$

from the dual. Note that by Claim 1, all the x_{ij}'s with a nonzero coefficient y_i in the left hand side of Equation (6.2) are distinct from the x_{ij}'s with a nonzero coefficient z_i on the right hand side of Equation (6.2). Suppose by way of contradiction that there are at most d nonzero entries in total in y and z. Fix one such nonzero entry, say, z_k. We can rewrite Equation (6.2) as:

$$z_k x_{kj} = \sum_{i=1}^{|P|} y_i x_{ij} - \sum_{i \neq k} z_i x_{ij}$$

and by dividing by z_k and relabeling the coefficients, we get:

$$x_{kj} = \sum_{i=1}^{|P|} \alpha_i x_{ij}$$

for some coefficients $\alpha_1, \ldots, \alpha_{|P|} \in \mathbb{R}$, where at most $d - 1$ of the α_i's are nonzero. But this says that there exist d points not in general position, which is a contradiction. Therefore Claim 2 is true. □

This completes the proof of Lemma 6.5. □

Now using Lemma 6.5, we can give the following upper bound for the size of the range set $\mathcal{R}$, in the one-slab problem on $\mathcal{U} = [m]^d$.

Lemma 6.6. *Given a grid $\mathcal{U} = [m]^d$, the size of the range set $\mathcal{R}$ consisting of all slabs is $O(m^{d^2+d})$.*

Proof. Based on Lemma 6.5, each slab on the grid $[m]^d$ can be determined by two parallel hyperplanes including $d + 1$ points. Thus we have:

$$\begin{aligned} |\mathcal{R}| = \sum_{k=1}^{d+1} \binom{|\mathcal{U}|}{k} \binom{|\mathcal{U}|}{d+1-k} &\leq \sum_{k=1}^{d+1} \binom{m^d}{k} \binom{m^d}{d+1-k} \\ &= \binom{2m^d}{d+1} = O(m^{d^2+d}) \end{aligned}$$

□

6.4.10 The Protocol

The protocol works as follows.

1. V processes the data stream as if for a RangeCount query with respect to $\mathcal{R}$, defined as the set of single slabs including $k + k' = d + 1$ points.
2. P returns a candidate slab S consisting of two parallel hyperplanes h_1, h_2, claimed as the slab with minimum width which covers all the points P in input data stream. P also sends a set T_1 of k points and a set T_2 of k' points claimed to satisfy the properties of Lemma 6.5.
3. V verifies that if $k + k' = d + 1$, checks that all points in T_1 lie on h_1 and all points in T_2 lie on h_2, and runs the PointQuery protocol $d + 1$ times to check that all points in $T_1 \cup T_2$ actually appeared in the input set P.
4. V initiates a RangeCount query for the range corresponding to the slab S, and verifies that the answer is $n = |P|$, i.e., that S covers all the input points.

Perfect completeness of the protocol is immediate from Lemma 6.5 and the completeness of the PointQuery and RangeCount protocols. The soundness error of the protocol is at most $(d + 2) \cdot \varepsilon_s$, where $\varepsilon_s \leq \frac{1}{3(d+2)}$ is an upper bound on the soundness errors of the PointQuery and RangeCount protocols. To see this, note that if T_1 and T_2 are as claimed, then there is no slab of width less than that of S covering the input points. And the probability that the verifier accepts when T_1 and T_2 are not as claimed is bounded by $(d + 2) \cdot \varepsilon_s$, via a union bound over all $(d + 1)$ invocations of the PointQuery protocol and the single invocation of the RangeCount protocol. Theorem 6.7 follows.

Theorem 6.7. *Given a stream of n input points from $\mathcal{U} = [m]^k$, there is a three-message SIP for verifying the width of the input with space and communication cost bounded by $O(d^4 \log^2 m)$.*

In the protocol of Theorem 6.7, the prover and verifier can be made to satisfy the same runtimes bounds as in the MEB protocol of Section 6.4.1, assuming the distinct function D satisfies the same "efficient computability" condition discussed there.

6.4.11 Verifying Approximate Metric k-Centers

Using the same ideas as for the MEB, we can verify a 2-*approximation* to the metric k-center problem via a three message SIP with polylogarithmic space and communication costs. At a high level, the SIP works as follows. Given a claimed solution, we reduce checking feasibility of the solution to a carefully constructed instance of the RangeCount problem, where the range space is defined as the set of all unions of k "balls" of radius r in the metric space. To verify that the solution is 2-approximate, we use the witness constructed by Gonzalez' [90] approximation algorithm: namely, $k+1$ points that are at least distance r apart (where r is the claimed 2-approximate radius). This sparse witness can be verified using the PointQuery protocol. Details follow.

6.4.12 Formalization of the Metric k-Center Problem

A k-center clustering of a set of points $p_1, \dots, p_n$ in a metric space (X, d) is a set of k centers $\mathcal{C} = \{c_1, \dots c_k\}$. The cost of such a clustering is

$$\text{cost}(C) = \max_i \min_j d(p_i, c_j).$$

Then we have the following definition.

Definition 6.8. *Let (X, d) be a metric space. Let $p_1, p_2, \dots, p_n, k$ be a stream of points from (X, d) followed by parameter k. An SIP computing a 2-approximation for the metric k-center problem with completeness error ε_c and soundness error ε_s has the following form. The prover begins the SIP by claiming that there exists a k-center clustering of cost r^*.*

- *If this claim is true, the verifier must accept with probability at least $1 - \varepsilon_c$.*
- *If there is no k-center clustering of cost at most $r^*/2$, the verifier must reject with probability at least $1 - \varepsilon_s$.*

It is easy to provide a protocol that works deterministically if the verifier is not required to process the input in a streaming manner. This is the standard 2-approximation algorithm of Gonzalez: the prover provides the following two proofs:

1. Proof of Feasibility: A set of *centers* $c_1, \dots, c_k$ satisfying $\max_i \min_j d(p_i, c_j) \leq r^*$ and
2. Proof of Approximate Optimality: A set of $k+1$ points $u_1, u_2, \dots, u_{k+1}$ from the stream with the promise that $\min_{i,j} d(u_i, u_j) \geq r^*$.

This guarantees a 2-approximation by the standard argument relying on the triangle inequality [90]. The verifier can easily check that the relevant conditions hold.

6.4.13 The SIP

Let $B_{d,r}(x) = \{y \in X \mid d(x,y) \leq r\}$ be a ball of radius r with center x in the metric space. We define a range space $\mathcal{R}$ consisting of all unions of k balls of radius r for all values of r:

$$\mathcal{R} = \{\cup_{z \in Z} B_{d,r}(z) \mid Z \subset X, |Z| = k, \exists x, y \in X, d(x,y) = r\}$$

Note that $|\mathcal{R}| = O(m^{k+2})$, where m is the size of metric space, i.e. $|X| = m$.

The protocol works as follows:

1. V processes the data stream as if for a RangeCount query using range space $\mathcal{R}$, as well as for $k+1$ parallel PointQuery queries.
2. P returns a candidate clustering $c_1, c_2, \ldots, c_k$ with the claimed cost r^*, as well as $k+1$ points $u_1, \ldots, u_{k+1}$ from the stream witnessing (approximate) optimality.
3. V initiates a RangeCount query for the range $\cup_{i=1}^{k} B_{d,r^*}(c_i)$ and verifies that the answer is $n = |P|$.
4. V verifies that the distance between all distinct pairs of points (u_i, u_j) is at least r^*, and invokes $(k+1)$ PointQuery queries to ensure that each u_i appeared in the input stream.

The correctness of the protocol follows from the correctness of Gonzalez's algorithm and Theorem 4.3. Note that approximating metric k-center to within a factor of $2 - \epsilon$ is NP-hard [76]. The above protocol is a streaming variant of an MA protocol. Under the widely-believed assumption that MA = NP, there is no $2 - \epsilon$ approximation for metric k-center with a polynomial-time verifier, regardless of whether the verifier processes the input in a streaming manner.

Theorem 6.9. *Let (X, d) be a metric space in which $|X| = m$. There is a streaming interactive protocol for verifying k-center clustering with space and communication complexity bounded by $O(k + \log(|\mathcal{R}|) \log(|\mathcal{R}| n))$, in which $|\mathcal{R}| \leq m^{k+2}$.*

As with our protocols for the MEB problem and computing the width of a point set, V and P can be made to run in quasilinear time if the metric d satisfies mild efficient-computability properties.

6.5 SIPs for General Clustering Problems

In this section, we give SIPs for two very general clustering problems: the k-center problem, and the k-slab problem. In the k-center problem, given a set of n points in $[m]^d$, the goal is to find k centers so as to minimize the maximum point-center distance. In the k-slab problem, the goal is instead to find k *hyperplanes* so as to minimize the maximum point-hyperplane distance. We gave a 3-message SIP for both problems in the special case where $k = 1$ in Section 6.4, as these cases are equivalent to the MEB and 1-slab problems. In order to handle the much harder case where $k \geq 2$ while keeping the costs polylogorithmic, we have to exploit sum-check techniques (cf. Section 4.2.6 and 4.3), which yields polylogarithmic-round SIPs.

6.5.1 k-Slabs

We first consider the k-slab problem. Even when $k = 2$, this problem appears to be difficult to solve efficiently without access to a prover: in fact, it was the first problem that does *not* admit a core set [97] (a small witness that yields a good approximation). The problem is therefore particularly ripe for outsourcing, as a computationally bounded algorithm is unlikely to solve it on worst case inputs, without access to a prover.

In the k-slab problem, it is convenient to think of each "cluster" as described not by a single hyperplane, but as the area between two parallel hyperplanes that contain all the points in that cluster (this was the viewpoint we took in the 1-slab protocol of Section 6.4). The *width* of this cluster is the distance between the two hyperplanes. We can now equivalently think of the k-slab objective as minimizing the maximum width of a cluster, and we will refer to this quantity also as the *width* of the k-slab. For any k-slab $\sigma \in \Re$, let $w(\sigma)$ denote this width.

6.5.2 Defining the Relevant Range Space

Each slab can be described by $d + 1$ points (that define the hyperplane) in $\mathcal{U} = [m]^d$ and a width parameter. A k-slab is a collection of k of such slabs. Let $\Re$ be the range space consisting set of all k-slabs. This range space has size $|\Re| = m^{kd^2+2kd}$. We will assume a canonical ordering of the ranges $\sigma_1, \sigma_2, \ldots,$ in increasing order of width (with an arbitrary ordering among ranges having the same width), as well as an effective enumeration procedure that given an index i returns the i^{th} range in the canonical order. We will also assume the existence of a mapping function $\mathcal{M} : w \to \{0, \ldots, |\Re| - 1\}$ which maps w to the smallest index i such that $w(\sigma_i) = w$. Notice that the verifier can compute this mapping function by explicit enumeration, using only enough space to store one range. This explicit enumeration requires extremely high runtime of up to $|\Re|$. However, it is possible to reduce the runtime of V to polylog$(|\Re|)$ per stream update using the techniques from [48] which we also stated in previous section.

6.5.3 Stream Observation Phase of the SIP

Let $P = (p_1, p_2, \ldots, p_n)$ be the stream of input points. As the verifier sees the data points, it generates a *derived stream* τ' as follows. For each point p_i in the actual input stream τ, V inserts into τ' all k-slabs $\sigma \in \Re$ which contain the point p_i. Notice that τ' is a deterministic function of τ, and hence the prover P, who sees τ, can also materialize τ', with no communication from V to P required to specify τ'. While the construction of this derived stream does not affect the space and communication costs of the protocol, it increases the verifier and prover running time dramatically. This can be avoided at the cost of a slight increase in communication. The details of this trick are described in [48]: the main idea is to observe that the frequency vector $f_{\mathbf{a}}$ is not arbitrary, since it tracks membership in ranges. This allows us to modify the extension polynomial used to report entries of the vector without needing to write down the explicit derived stream. By design, is that the frequency f_σ of the range σ in this derived stream τ' is the number of points that σ contains: $f_\sigma = |\sigma \cap P|$.

6.5.4 Proving Feasibility

After the stream τ has passed, the prover supplies a candidate k-slab σ^* and claims that this has optimal width $w^* = w(\sigma^*)$. By applying the $\mathsf{RangeCount}$ protocol from Theorem 4.3 to the derived stream τ', the verifier can check that $f^*_\sigma = n$ and is therefore feasible. This feasibility check requires only 3 messages. It is the optimality check below that requires a super constant number of rounds.

6.5.5 Proving Optimality

The verifier must now check if the optimal width is w. Given a subset $S \subseteq \Re$ of k-slabs, let $\mathbb{I}_S \colon \{0,1\}^{\log|\Re|} \to \{0,1\}$ denote the indicator function that evaluates to 1 on the binary representation a range σ of a k-slab if $\sigma \in S$, and evaluates to 0 otherwise. Let $S := \{\sigma \colon w(\sigma) < w^*\}$, and let $T = \{\sigma \colon f_\sigma = n\}$. Let $F = \sum_{\sigma \in \Re} \mathbb{I}_S(\sigma)\mathbb{I}_T(\sigma)$. Then the prover has supplied an optimal range σ^* if and only if $F = 0$.

Let $\mathbb{F}$ be a field of prime order satisfying $6n^2 \leq |\mathbb{F}| \leq 6n^3$. Let $\hat{\mathbb{I}}_S \colon \mathbb{F}^{\log|\Re|} \to \mathbb{F}$ be the multilinear extension of $\mathbb{I}_S$, and let $\hat{\mathbb{I}}_T$ be the multilinear extension of $\mathbb{I}_T$. That is, $\hat{\mathbb{I}}_S$ is the unique multilinear polynomial over $\mathbb{F}$ satisfying $\hat{\mathbb{I}}_S(\sigma) = \mathbb{I}_S(\sigma)$ for all $\sigma \in \{0,1\}^P \log|\Re|$, and similarly for $\hat{\mathbb{I}}_T$. It is standard that

$$\hat{\mathbb{I}}_S = \sum_{\sigma \in \{0,1\}^{\log \Re}} \mathbb{I}_S(\sigma) \cdot \chi_\sigma, \tag{6.3}$$

where

$$\chi_\sigma(x_1, \ldots, x_{\log|\Re|}) := \prod_{i=1}^{\log|\Re|} (x_1\sigma_i + (1-x_i)(1-\sigma_i)), \tag{6.4}$$

and similarly for $\hat{\mathbb{I}}_T$. To compute F, it suffices to apply the sum-check protocol to the polynomial $g := \hat{\mathbb{I}}_S \cdot \hat{\mathbb{I}}_T$. The protocol requires $\log|\Re|$ rounds, and the total communication cost is $O(\log|\Re|)$ field elements. To perform the necessary check in the final round of this protocol, V needs to evaluate g at a random point $\mathbf{r} \in \mathbb{F}^{\log|\Re|}$. By definition of g, it suffices for V to evaluate $\hat{\mathbb{I}}_T(\mathbf{r})$ and $\hat{\mathbb{I}}_S(\mathbf{r})$. Since the set S does not depend on the stream (S depends only on the claimed optimal width w^*), V can evaluate $\hat{\mathbb{I}}_S(\mathbf{r})$ *after* the stream has passed, using $O(\log(|\Re|) \cdot \log|\mathbb{F}|)$ bits of space, using standard techniques (see for example [66, Section 2]). However, it is not possible for V to evaluate $\hat{\mathbb{I}}_T(\mathbf{r})$ in a streaming manner. Instead, V asks P to *tell her* $\hat{\mathbb{I}}_T(\mathbf{r})$, and checks that $\hat{\mathbb{I}}_T(\mathbf{r})$ by invoking the streaming implementation of the GKR protocol (cf. Lemma 4.3). More precisely, similar to [63, Section 3.3], we observe that Fermat's Little Theorem implies that $f_\sigma = n$ if and only if $(f_\sigma - n)^{|\mathbb{F}|-1} \equiv 1 \mod |\mathbb{F}|$. This implies via Equation (6.3) that $\mathbb{I}_T(\mathbf{r}) = \sum_{\sigma \in \{0,1\}^{\log|\Re|}} (f_\sigma - n)^{|\mathbb{F}|-1} \cdot \chi_\sigma(\mathbf{r})$, where χ_σ was defined in Equation (6.4). As in [63, Section 3.3], it is possible to compute the right hand side of this equality by an arithmetic circuit $\mathcal{C}$ of size $O(|\Re|)$ and depth $O(\log|\mathbb{F}|) = O(\log n)$ over $\mathbb{F}$. By applying the GKR protocol to $\mathcal{C}$, V forces P to faithfully provide $\hat{\mathbb{I}}_T(\mathbf{r})$. This completes the protocol. Completeness and soundness follow from completeness and soundness of the sum-check protocol and of the GKR protocol.

6.5.6 Protocol Costs

The total communication cost of the protocol $O(\log(n) \cdot \log(|\Re|) \cdot \log|\mathbb{F}|) = O(k \cdot d^2 \cdot \log(m) \cdot \log^2 n)$ bits. The total space cost is $O(\log(|\Re|) \cdot \log(|\mathbb{F}|)) = O(k \cdot d^2 \cdot \log(m) \cdot \log(n))$ bits. The total number of rounds required is $O(\log(n) \cdot \log(|\Re|)) = O(k \cdot d^2 \cdot \log(m) \cdot \log(n))$.

Theorem 6.10. *Given a stream of n points, there is a streaming interactive proof for computing the optimal k-slab, with space and communication bounded by $O(k \cdot d^2 \cdot \log(m) \cdot \log^2 n)$. The total number of rounds is $O(k \cdot d^2 \cdot \log(m) \cdot \log(n))$.*

We remark that it is possible to reduce the number of rounds in Theorem 6.10 by a factor of $\log(n)$, using a technique introduced by Gur and Raz [95], and applied by Klauck and Prakash [120] to obtain an $O(\log|\Re|)$-round SIP for computing the number of distinct items in a data stream. However, these techniques sacrifice perfect completeness, and increase the communication complexity of the protocol by polylogarithmic factors. We omit the details of this technique for brevity.

6.5.7 k-Center

We can use the same idea as in Section 6.5.1 to verify solutions for Euclidean k-center. The primary difference in the proof is in defining the relevant range space, which here consists of unions of k balls of radius r, for all choices of centers and radii in the grid. The size of this range space is m^{2kd}, compared to m^{kd^2+2kd} in the case of the k-slab problem. Here we provide k-center verification protocol in more details:

Here the notion of "cluster" is described by a ball centered at one of the input points with some certain *radius* which contains all the points in that cluster. Equivalently, the k-center objective is minimizing the maximum radius of a cluster, referred as the radius of k-center and for any k-center $\sigma \in \mathcal{B}^k$, let $r(\sigma)$ denote this radius.

6.5.8 Range Space

Let $\mathcal{B} = \{B(c, j\epsilon) : c \in \mathcal{U} = [m]^d, j \in \mathbb{Z}, 0 \leq j \leq \frac{1}{\epsilon}\}$ be the set of all balls of radius between 0 and 1 (which is quantized by ϵ). A k-center is a collection of k of such balls. Let $\mathcal{B}^k$ be the set of all such ranges. This range space has size $|\mathcal{B}^k| = m^{2kd}$. As before, we will assume a canonical ordering of the ranges $\sigma_1, \sigma_2, \ldots$, in increasing order of radius and the existence of a mapping function $\mathcal{M} : r \to [|\mathcal{B}^k|]$ which maps r to the smallest index i such that $r(\sigma_i) = r$, and is efficiently computable by the verifier.

6.5.9 Preprocessing

The verifier generate a derived stream from the input P (as described before) which the frequencies are computed over that and the frequency f_σ of the range σ in this derived stream is the number of points it contains: $f_\sigma = |\sigma \cap P|$.

6.5.10 Feasibility

After the stream has passed, the prover supplies a candidate k-center σ^* and claims that this has optimal radius $r^* = r(\sigma^*)$. Using the RangeCount protocol from Theorem 4.3 the verifier can check that $f^*_\sigma = n$ and is therefore feasible.

6.5.11 Optimality

The verifier must now check if the optimal radius of k-center clustering is in fact r^*. Following the analysis in previous section for k-slab, we can re-express the certificate of optimality in k-center in a similar manner as: r^* is the optimal radius if and only if $F' = \mathcal{M}(w^*) - 1$, in which :

$$F'(r^*) = \sum_{\sigma \in \mathcal{B}^k} \mathbb{I}(r(\sigma) < r^*) \mathbb{I}(f'_\sigma > 0)$$

$$\mathcal{M}(r^*) - 1 \; = \sum_{\sigma \in \mathcal{B}^k} \mathbb{I}(r(\sigma) < r^*)$$

Accordingly, the problem reduces to the prefix distinct elements (F_0) again which was studied in detail in the last section. This way we obtain a streaming interactive protocol for checking the feasibility and optimality of Euclidean k-center clustering problem (similar to what we presented for k-slab), but with different range space. We skip restating the protocol here and just summarize the main result:

Theorem 6.11. *Given a stream of n input points, there is an SIP for computing the optimal k-center with space and communication bounded by $O(k \cdot d \cdot \log(m) \cdot \log^2 n)$. The total number of rounds is $O(k \cdot d \cdot \log(m) \cdot \log(n))$.*

CHAPTER 7

SUMMARY AND FUTURE DIRECTIONS

In this dissertation, we developed novel sublinear algorithms for various optimization and data analysis problems including data summarization and coreset, clustering, matrix problems and massive graphs.

In Chapter 2, we presented the first results for constructing small size coresets with theoretical guarantees for various types of range counting queries on uncertain data, modeled by the indecisive locational uncertainty, where each uncertain point has a probability density describing its locations described as k distinct locations. These can be essential tools for monitoring a subset of a large noisy data set, as a way to approximately monitor the full uncertainty. The summary of results can be found in Table 7.1. There are many future directions on this topic, in addition to tightening the provided bounds especially for other range spaces. Can we remove the dependence on k without random sampling? How can we generalize our results and techniques to the family of continuous distributions over the uncertain data? Another interesting direction is to extend the problem of computing coreset over uncertain data to streaming settings. After our work, Munteanu, Sohler and Feldman [134] presented a polynomial time approximate algorithm for the probabilistic enclosing ball problem with extension to streaming setting which uses coreset techniques. We hope that the new techniques and ideas presented in this dissertation open new directions for generalizing the existing lines of research in shape fitting, clustering and machine learning problems to their probabilistic versions.

In Chapter 3, we presented an algorithm for constructing a sublinear size coreset on massive graphs to compute a $(1+\epsilon)$-approximation to maxcut size and objective function for correlation clustering. This algorithm is based on a biased sampling procedure on the vertices and edges. The vertex sampling obtain a smaller subgraph with sublinear size set of vertices, which can be used as a coreset with approximation guarantee on maxcut size. Given this small coreset, then an edge sampling algorithm can be used to sparsify the resulted subgraph further by sampling the edges and obtain a sublinear size subgraph (with regards to vertices and edges), used as a sparse coreset for obtaining a $(1+\epsilon)$-approximation to maxcut size and correlation clustering objective value. The summary of related results is presented in Table 7.2. There are several future research directions for our work: First, the core set construction algorithms presented here can be adapted to produce 2-pass *streaming* algorithms that use $\tilde{O}(n^{1-\delta})$ space and yield $(1-\epsilon)$-approximations for both problems. The main idea is to use the properties of CountMin sketch [65] and ℓ_1-sampling [17, 106, 133] to obtain the coreset resulted from biased vertex sampling procedure. In the case of maxcut, this would be the first example of obtaining a sublinear algorithm without relying either on large density or strong regularity assumptions. In the case of correlation clustering, this improves the semistreaming algorithms (space $\tilde{O}(n)$) proposed by Ahn et al. [9], albeit using an extra pass. Furthermore, our algorithm is based on a biased sampling procedure with regard to degree distributions and edge weights. One crucial question here is whether we can use instead an uniform random sampling algorithm for constructing coreset while obtaining the same approximation guarantees? Note that in [77], it is shown that for special graphs where the input is near-regular (i.e., $d_i \leq c \cdot d_{average}$), uniform random sampling on vertices obtains a coreset for computing a $(1+\epsilon)$-approximation to maxcut size, but it seems that the technical proofs presented in [77] rely heavily on the near-regularity assumptions and cannot be easily adapted to the graphs with general degree distributions. The other interesting direction is to explore if the ideas presented in our work can be used for constructing a small size coreset or designing a sublinear space steaming algorithm for other graph problems. We expect ideas from our analysis (for the behavior of random induced subprograms of linear programs) to be applicable to other settings as well, especially ones for which the framework of [22] is applicable. An example that we have not presented is the so-called *cut norm* problem (which is easier than the so-called 'Max Cut-Gain', which is related to maxcut). Here, our methods give additive $\varepsilon n\Delta$ (where Δ is the average degree) approximations, which, for appropriate parameter choices, can be much better than results obtained via more analytic methods [143].

In Chapter 4, we gave an overview of basic known tools and techniques used for designing our streaming verification algorithms.

In Chapter 5, we presented efficient streaming interactive proofs for exact verification of graph matching (bipartite, nonbipartite and weighted). We also designed SIPs for metric TSP, exact triangle counting, number of connected components, bipartiteness minimum spanning tree and connectivity. These protocols have polylog space and sublinear (in number of edges) communication complexity. The protocols presented for weighted matching and metric TSP are the first results in any streaming verification model. Interestingly, our work shows in several cases an exponential improvement in the product of space and communication versus a single round of communication in annotation model (for which an $\Omega(n^2)$ lower bound exists), showing that while a prover can help, it needs to have an extended conversation with the verifier. The summary of related results is presented in Table 7.3. Our matching protocol requires the prover to send back an actual matching and a certificate for it. Suppose we merely wanted to verify a claimed cost for the matching. Is there a way to verify this with less communication? Another interesting question is to consider designing SIPs for graph problems which are known to be NP-hard or the ones with known space lower bounds in streaming model. For example, is there any efficient SIP for verifying Max Cut in streaming graphs? In our SIPs for matching, we assume that the edge weight updates are atomic. Can we relax this constraint? Justin Thaler [148] observed that by using techniques from [64], we can design a SIP with $\log^2 n$ space cost and $O(W(\log W + \log n) \log n)$ bits of communication, in which W is the upper bound on the edge weights (i.e. $w_{ij} \leq W$). Both protocols will result in similar costs for any instance where the edge weights are at most $O(n)$, while having the advantage of handling incrementally-specified edge weights in the second approach. However, in the general case where $w_{ij} \in [n^c]$, our solution still has lower communication cost (worst case $n \log n$).

Finally, in Chapter 6 we initiated the study of streaming interactive proofs for problems in data analysis and presented efficient SIPs for various fundamental problems such as clustering, shape fitting and matrix analysis. The related results are summarized in Table 7.4. As an interesting future research direction, we can explore outsourcing computations for solving large scale machine learning problems, which necessitate designing efficient verification protocols. For example, how we can do verification for fundamental learning tasks such as regression, classification and more complicated clustering tasks? This requires developing new tools and techniques in the field of streaming verification.

Table 7.1: Our Results for coreset size for different range queries. In random sampling, v is the VC-dimension of the associated range space with the corresponding family of range $\mathcal{R}$. In discrepancy-based approach the corresponding range $\mathcal{R}$ is defined as axis-aligned rectangles. For RQ queries, the value $\alpha_{\epsilon,t} = \epsilon + \sqrt{(\frac{1}{2t})\log(\frac{2}{\epsilon})}$, in which t is the size of coreset.

Problem	Random Sampling	Discrepancy-based Approach
ϵ-RE coreset	$O((\frac{1}{\epsilon^2})(v+\log(\frac{k}{\delta})))$	$O((\frac{\sqrt{k}}{\epsilon})\cdot\log^{\frac{3d-1}{2}}(\frac{k}{\epsilon}))$
ϵ-RC coreset	$O((\frac{1}{\epsilon^2})(v+\log(\frac{1}{\delta})))$	$O((\frac{k^{3d+\frac{1}{2}}}{\epsilon})\log^{6d-\frac{1}{2}}(\frac{k}{\epsilon}))$
$(\epsilon,\alpha_{\epsilon,t})$-RQ coreset	$O((\frac{1}{\epsilon^2})(v+\log(\frac{k}{\delta})))$	$O((\frac{\sqrt{k}}{\epsilon})\cdot\log^{\frac{3d-1}{2}}(\frac{k}{\epsilon}))$

Table 7.2: Our results on coreset size for computing $(1+\epsilon)$-approximation to maxcut size and correlation clustering objective value. The input graph has $O(n^{1+\delta})$ edges, where $0 \leq \delta \leq 1$ and average degree $\Delta = n^{\delta}$. The ϵ-coreset is obtained by biased vertex sampling algorithm and sparse ϵ-coreset is resulted from applying edge sampling algorithm on ϵ-coreset.

Maxcut/ Correlation Clustering	Space Complexity
ϵ-coreset	$\tilde{O}(n^{1-\delta})$ vertex size
sparse ϵ-coreset	$\tilde{O}(n^{1-\delta})$

Table 7.3: Our results on streaming graph verification problems. All bounds expressed in bits, upto constant factors. For the matching results, $\rho = \min(n, C)$ where C is the cost of the optimal matching (weighted or unweighted). Note that for the MST, the verification is for a $(1+\epsilon)$-approximation. For the TSP, the verification is for a $(3/2+\varepsilon)$-approximation. Parameter γ' is a linear function of γ and is strictly more than 1 as long as γ is a sufficiently large constant.

	$\log n$ rounds		$\gamma = O(1)$ rounds	
Problem	Verifier Space	Communication	Verifier Space	Communication
Triangle Count	$\log^2 n$	$\log^2 n$	$\log n$	$n^{1/\gamma}\log n$
Matchings	$\log^2 n$	$(\rho+\log n)\log n$	$\log n$	$(\rho+n^{1/\gamma'})\log n$
Connectivity	$\log^2 n$	$n\log n$	$\log n$	$n\log n$
MST*	$\log^2 n$	$n\log^2 n/\varepsilon$	$\log n$	$n\log^2 n/\varepsilon$
TSP*	$\log^2 n$	$n\log^2 n/\varepsilon$	$\log n$	$n\log^2 n/\varepsilon$

Table 7.4: Results for geometric and data analysis problems. In the Euclidean space the points are taken from the universe $\mathcal{U}[m]$. In metric space, the points are taken from (X, d) in which d is the space dimension and $|X| = m$ and $|\mathcal{R}| \leq m^{k+2}$. In matrix related problems, (h, v) is any pair of positive integers such that $h, v \geq n$. Also the cost of annotation protocols is described as a pair of values for (space, communication) complexity. In matrix multiplication A is a $k \times n$ matrix and B is an $n \times k$ matrix and the goal is to verify the matrix product $C = A \cdot B$. In eigenstructure, the goal is to verify the collection of k eigenpairs $(\lambda_i, \mathbf{v}_i)$ are orthogonal, and each satisfy $||A\mathbf{v}_i - \lambda_i \mathbf{v}_i|| \leq \epsilon$.

Problem	Cost (Communication/Space)	Rounds of Interaction
Minimum Enclosing Ball	$O(d^2 \cdot \log^2 m)$	3-messages SIP
Width	$O(d^4 \cdot \log^2 m)$	3-messages SIP
Metric k-Centers	$O(k + \log(|\Re|) \cdot \log(n \cdot |\Re|))$	3-messages SIP
k-slab covering	$O(k \cdot d^2 \cdot \log m \cdot \log^2 n)$	$k \cdot d^2 \cdot \log m \cdot \log n$
Euclidean k-center	$O(k \cdot d \cdot \log m \cdot \log^2 n)$	$k \cdot d \cdot \log m \cdot \log n$
Matrix Multiplications	$(v \log n, k \cdot k' \cdot h \log n)$	Annotation model
Matrix Eigenstructure	$O(v \cdot \log(\frac{n}{\epsilon}), k^2 \cdot h \cdot \log(\frac{n}{\epsilon}))$	Annotation model

REFERENCES

[1] Abdullah, A., Daruki, S., and Phillips, J. M. Range counting coresets for uncertain data. In *Proceedings of the Twenty-ninth Annual Symposium on Computational Geometry* (2013), ACM, pp. 223–232.

[2] Abdullah, A., Daruki, S., Roy, C. D., and Venkatasubramanian, S. Streaming verification of graph properties. In *27th International Symposium on Algorithms and Computation (ISAAC 2016)* (2016), vol. 64, Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, pp. 3:1–3:14.

[3] Agarwal, P. K., Cheng, S.-W., Tao, Y., and Yi, K. Indexing uncertain data. In *PODS* (2009).

[4] Agarwal, P. K., Har-Peled, S., and Varadarajan, K. Geometric approximations via coresets. *Current Trends in Combinatorial and Computational Geometry (E. Welzl, ed.)* (2007).

[5] Agarwal, P. K., Har-Peled, S., and Varadarajan, K. R. Approximating extent measure of points. *Journal of ACM 51*, 4 (2004), 2004.

[6] Agarwal, P. K., Har-Peled, S., and Varadarajan, K. R. Geometric approximation via coresets. *Combinatorial and Computational Geometry 52* (2005), 1–30.

[7] Agarwal, P. K., and Sharathkumar, R. Streaming algorithms for extent problems in high dimensions. In *Proc 21st SODA* (2010), pp. 1481–1489.

[8] Agrawal, P., Benjelloun, O., Sarma, A. D., Hayworth, C., Nabar, S., Sugihara, T., and Widom, J. Trio: A system for data, uncertainty, and lineage. In *PODS* (2006).

[9] Ahn, K., Cormode, G., Guha, S., McGregor, A., and Wirth, A. Correlation clustering in data streams. In *International Conference on Machine Learning* (2015), pp. 2237–2246.

[10] Ahn, K. J., and Guha, S. Graph sparsification in the semi-streaming model. In *International Colloquium on Automata, Languages, and Programming* (2009), Springer, pp. 328–338.

[11] Ahn, K. J., and Guha, S. Access to data and number of iterations: Dual primal algorithms for maximum matching under resource constraints. *arXiv preprint arXiv:1307.4359* (2013).

[12] Ahn, K. J., and Guha, S. Linear programming in the semi-streaming model with application to the maximum matching problem. *Information and Computation 222* (2013), 59–79.

[13] Ahn, K. J., Guha, S., and McGregor, A. Analyzing graph structure via linear measurements. In *SODA* (2012), SIAM, pp. 459–467.

[14] Ahn, K. J., Guha, S., and McGregor, A. Graph sketches: Sparsification, spanners, and subgraphs. In *Proceedings of the 31st Symposium on Principles of Database Systems* (2012), ACM, pp. 5–14.

[15] Ailon, N., and Karnin, Z. A note on: No need to choose: How to get both a PTAS and sublinear query complexity. *arXiv preprint arXiv:1204.6588* (2012).

[16] Alon, N., De La Vega, W. F., Kannan, R., and Karpinski, M. Random sampling and approximation of max-CSP problems. In *Proceedings of the Thiry-fourth Annual ACM Symposium on Theory of Computing* (2002), ACM, pp. 232–239.

[17] Andoni, A., Krauthgamer, R., and Onak, K. Streaming algorithms via precision sampling. In *Foundations of Computer Science (FOCS), 2011 IEEE 52nd Annual Symposium on* (2011), IEEE, pp. 363–372.

[18] Andoni, A., Krauthgamer, R., and Woodruff, D. P. The sketching complexity of graph cuts. *arXiv preprint arXiv:1403.7058* (2014).

[19] Andoni, A., and Nguyen, H. Eigenvalues of a matrix in the streaming model. In *Proc. 24th SODA* (2013), pp. 1729–1737.

[20] Anthony, M., and Bartlett, P. L. *Neural network learning: theoretical foundations*. Cambridge University Press, 1999.

[21] Arora, S., and Barak, B. *Computational complexity: A modern approach*. Cambridge University Press, 2009.

[22] Arora, S., Karger, D., and Karpinski, M. Polynomial time approximation schemes for dense instances of NP-hard problems. In *Proceedings of the Twenty-seventh Annual ACM Symposium on Theory of Computing* (1995), ACM, pp. 284–293.

[23] Assadi, S., Khanna, S., Li, Y., and Yaroslavtsev, G. Tight bounds for linear sketches of approximate matchings. *arXiv preprint arXiv:1505.01467* (2015).

[24] Azar, P. D., and Micali, S. Rational proofs. In *STOC* (2012), ACM, pp. 1017–1028.

[25] Babai, L., Frankl, P., and Simon, J. Complexity classes in communication complexity theory. In *Proc. 27th IEEE FOCS* (1986), pp. 337–347.

[26] Badoiu, M., and Clarkson, K. L. Smaller core-sets for balls. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms* (2003), Society for Industrial and Applied Mathematics, pp. 801–802.

[27] Bădoiu, M., Har-Peled, S., and Indyk, P. Approximate clustering via core-sets. In *STOC* (2002).

[28] Bahmani, B., Kumar, R., and Vassilvitskii, S. Densest subgraph in streaming and mapreduce. *Proceedings of the VLDB Endowment 5*, 5 (2012), 454–465.

[29] Bandyopadhyay, D., and Snoeyink, J. Almost-delaunay simplices: Nearest neighbor relations for imprecise points. In *SODA* (2004).

[30] Bansal, N., Blum, A., and Chawla, S. Correlation clustering. *Machine Learning 56*, 1-3 (2004), 89–113.

[31] Bar-Yossef, Z. *The Complexity of Massive Data Set Computations*. PhD thesis, University of California at Berkeley, 2002.

[32] Bar-Yossef, Z., Kumar, R., and Sivakumar, D. Reductions in streaming algorithms, with an application to counting triangles in graphs. In *SODA* (2002), Society for Industrial and Applied Mathematics, pp. 623–632.

[33] Barak, B., Hardt, M., Holenstein, T., and Steurer, D. Subsampling mathematical relaxations and average-case complexity. In *Proceedings of the Twenty-second Annual ACM-SIAM Symposium on Discrete Algorithms* (Philadelphia, PA, USA, 2011), SODA '11, Society for Industrial and Applied Mathematics, pp. 512–531.

[34] Beck, J. Roth's estimate of the discrepancy of integer sequences is nearly sharp. *Atorica 1* (1981), 319–325.

[35] Berge, C. Sur le couplage maximum d'un graphe. *Comptes Rendus Hebdomadaires Des Séances de l'Académie des Sciences 247* (1958), 258–259.

[36] Bhaskara, A., Daruki, S., and Venkatasubramanian, S. Sublinear algorithms for maxcut and correlation clustering. *arXiv preprint arXiv:1802.06992* (2018).

[37] Bohus, G. On the discrepancy of 3 permutations. *Random Structures and Algorithms 1*, 2 (1990), 215–220.

[38] Boucheron, S., Lugosi, G., and Massart, P. Concentration inequalities using the entropy method. *Annals of Probability* (2003), 1583–1614.

[39] Braverman, V., Ostrovsky, R., and Vilenchik, D. How hard is counting triangles in the streaming model? In *Automata, Languages, and Programming*. Springer, 2013, pp. 244–254.

[40] Bdoiu, M., and Clarkson, K. Smaller core-sets for balls. In *SODA* (2003).

[41] Burdick, D., Deshpande, P. M., Jayram, T., Ramakrishnan, R., and Vaithyanathan, S. OLAP over uncertain and imprecise data. In *VLDB* (2005).

[42] Buriol, L. S., Frahling, G., Leonardi, S., Marchetti-Spaccamela, A., and Sohler, C. Counting triangles in data streams. In *Proceedings of the Twenty-fifth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems* (2006), ACM, pp. 253–262.

[43] Bury, M., and Schwiegelshohn, C. Sublinear estimation of weighted matchings in dynamic data streams. *arXiv preprint arXiv:1505.02019* (2015).

[44] Chakrabarti, A., Cormode, G., Goyal, N., and Thaler, J. Annotations for sparse data streams. In *SODA'14* (2014), pp. 687–706.

[45] Chakrabarti, A., Cormode, G., and McGregor, A. Annotations in data streams. In *Automata, Languages and Programming*. Springer, 2009, pp. 222–234.

[46] Chakrabarti, A., Cormode, G., and McGregor, A. Annotations in data streams. In *Automata, Languages and Programming*. Springer, 2009, pp. 222–234.

[47] Chakrabarti, A., Cormode, G., McGregor, A., and Thaler, J. Annotations in data streams. *ACM Transactions on Algorithms (TALG) 11*, 1 (2014), 7.

[48] Chakrabarti, A., Cormode, G., McGregor, A., Thaler, J., and Venkatasubramanian, S. On interactivity in Arthur-Merlin communication and stream computation. In *Proc. IEEE Conference on Computational Complexity* (2015).

[49] Chakrabarti, A., Cormode, G., McGregor, A., Thaler, J., and Venkatasubramanian, S. Verifiable stream computation and Arthur–Merlin communication. In *CCC* (2015).

[50] Chan, T. M. Faster core-set constructions and data-stream algorithms in fixed dimensions. *Computational Geometry 35*, 1 (2006), 20–35.

[51] Chazelle, B. *The discrepancy method*. Cambridge, 2000.

[52] Chazelle, B., and Matousek, J. On linear-time deterministic algorithms for optimization problems in fixed dimensions. *Journal of Algorithms 21* (1996), 579–597.

[53] Chazelle, B., Rubinfeld, R., and Trevisan, L. Approximating the minimum spanning tree weight in sublinear time. *SIAM Journal on Computing 34*, 6 (2005), 1370–1379.

[54] Chazelle, B., and Welzl, E. Quasi-optimal range searching in spaces of finite VC-dimension. *Discrete and Computational Geometry 4* (1989), 467–489.

[55] Chen, J., McCauley, S., and Singh, S. Rational proofs with multiple provers. *CoRR abs/1504.08361* (2015).

[56] Chen, K. On coresets for k-median and k-means clustering in metric and euclidean spaces and their applications. *SIAM Journal on Computing 39*, 3 (2009), 923–947.

[57] Cheng, R., Xia, Y., Prabhakar, S., Shah, R., and Vitter, J. S. Efficient indexing methods for probabilistic threshold queries over uncertain data. In *VLDB* (2004).

[58] Chierichetti, F., Dalvi, N., and Kumar, R. Correlation clustering in mapreduce. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (2014), ACM, pp. 641–650.

[59] Chitnis, R., Cormode, G., Hajiaghayi, M., and Monemizadeh, M. Parameterized streaming: Maximal matching and vertex cover. In *SODA* (2015), SIAM, pp. 1234–1251.

[60] Cormode, G. Bertinoro workshop 2011, problem 47. `http://sublinear.info/index.php?title=Open_Problems:47`.

[61] Cormode, G., and Garafalakis, M. Histograms and wavelets of probabilitic data. In *ICDE* (2009).

[62] Cormode, G., Li, F., and Yi, K. Semantics of ranking queries for probabilistic data and expected ranks. In *ICDE* (2009).

[63] Cormode, G., Mitzenmacher, M., and Thaler, J. Practical verified computation with streaming interactive proofs. In *Proc. 3rd ITCS* (2012), ACM, pp. 90–112.

[64] Cormode, G., Mitzenmacher, M., and Thaler, J. Streaming graph computations with a helpful advisor. *Algorithmica 65*, 2 (2013), 409–442.

[65] Cormode, G., and Muthukrishnan, S. An improved data stream summary: The count-min sketch and its applications. *Journal of Algorithms 55*, 1 (2005), 58–75.

[66] Cormode, G., Thaler, J., and Yi, K. Verifying computations with streaming interactive proofs. *Proceedings of the VLDB Endowment 5*, 1 (2011), 25–36.

[67] Crouch, M., and Stubbs, D. M. Improved streaming algorithms for weighted matching, via unweighted matching. *APPROX/RANDOM 28* (2014), 96–104.

[68] Cunningham, W., and Marsh, A. A primal algorithm for optimum matching. In *Polyhedral Combinatorics*. Springer, 1978, pp. 50–72.

[69] Dalvi, N., and Suciu, D. Efficient query evaluation on probabilistic databases. In *VLDB* (2004).

[70] Daruki, S., Thaler, J., and Venkatasubramanian, S. Streaming verification in data analysis. In *Algorithms and Computation*. Springer Berlin Heidelberg, 2015, pp. 715–726.

[71] Dasgupta, A., Drineas, P., Harb, B., Kumar, R., and Mahoney, M. W. Sampling algorithms and coresets for Lp regression. *SIAM Journal on Computing 38*, 5 (2009), 2060–2078.

[72] Dubhashi, D. P., and Panconesi, A. *Concentration of Measure for the Analysis of Randomized Algorithms*. Cambridge University Press, 2009.

[73] Edwards, M., and Varadarajan, K. No coreset, no cry: II. In *Proc. FSTTCS* (2005), pp. 107–115.

[74] Epstein, L., Levin, A., Mestre, J., and Segev, D. Improved approximation guarantees for weighted matching in the semi-streaming model. *SIAM Journal on Discrete Mathematics 25*, 3 (2011), 1251–1265.

[75] Esfandiari, H., Hajiaghayi, M. T., Liaghat, V., Monemizadeh, M., and Onak, K. Streaming algorithms for estimating the matching size in planar graphs and beyond. In *SODA* (2015), SIAM, pp. 1217–1233.

[76] Feder, T., and Greene, D. Optimal algorithms for approximate clustering. In *Proc. ACM STOC* (1988), ACM, pp. 434–444.

[77] Feige, U., and Schechtman, G. On the optimality of the random hyperplane rounding technique for max cut. *Random Structures & Algorithms 20*, 3 (2002), 403–440.

[78] Feldman, D., Monemizadeh, M., Sohler, C., and Woodruff, D. P. Coresets and sketches for high dimensional subspace approximation problems. In *Proceedings of the Twenty-first Annual ACM-SIAM Symposium on Discrete Algorithms* (2010), Society for Industrial and Applied Mathematics, pp. 630–649.

[79] Feldman, D., Schmidt, M., and Sohler, C. Turning big data into tiny data: Constant-size coresets for k-means, PCA and projective clustering. In *Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms* (2013), SIAM, pp. 1434–1453.

[80] Fenner, S. A., Gurjar, R., and Thierauf, T. Bipartite perfect matching is in quasi-NC. *Electronic Colloquium on Computational Complexity (ECCC) 22* (2015), 177.

[81] Fotakis, D., Lampis, M., and Paschos, V. T. Sub-exponential approximation schemes for CSPs: From dense to almost sparse. *arXiv preprint arXiv:1507.04391* (2015).

[82] Grtner, B., and Jaggi, M. Coresets for polytope distance. In *Proceedings of the Twenty-fifth Annual Symposium on Computational Geometry* (2009), ACM, pp. 33–42.

[83] Giotis, I., and Guruswami, V. Correlation clustering with a fixed number of clusters. In *Proceedings of the Seventeenth Annual ACM-SIAM Symposium on Discrete Algorithm* (2006), Society for Industrial and Applied Mathematics, pp. 1167–1176.

[84] Goel, A., Indyk, P., and Varadarajan, K. R. Reductions among high dimensional proximity problems. In *Proc. 12th SODA* (2001), S. R. Kosaraju, Ed., pp. 769–778.

[85] Goel, A., Kapralov, M., and Khanna, S. Graph sparsification via refinement sampling. *arXiv preprint arXiv:1004.4915* (2010).

[86] Goel, A., Kapralov, M., and Post, I. Single pass sparsification in the streaming model with edge deletions. *arXiv preprint arXiv:1203.4900* (2012).

[87] Goldreich, O. *Introduction to property testing*. Cambridge University Press, 2017. `http://www.wisdom.weizmann.ac.il/~oded/pt-intro.html`.

[88] Goldreich, O., Goldwasser, S., and Ron, D. Property testing and its connection to learning and approximation. *Journal of the ACM (JACM) 45*, 4 (1998), 653–750.

[89] Goldwasser, S., Kalai, Y. T., and Rothblum, G. N. Delegating computation: Interactive proofs for muggles. *Journal of the ACM (JACM) 62*, 4 (2015), 27.

[90] Gonzalez, T. F. Clustering to minimize the maximum intercluster distance. *Theoretical Computer Science 38* (1985), 293–306.

[91] Guibas, L. J., Salesin, D., and Stolfi, J. Epsilon geometry: Building robust algorithms from imprecise computations. In *SoCG* (1989).

[92] Guibas, L. J., Salesin, D., and Stolfi, J. Constructing strongly convex approximate hulls with inaccurate primitives. *Algorithmica 9* (1993), 534–560.

[93] Guo, S., Hubcek, P., Rosen, A., and Vald, M. Rational arguments: Single round delegation with sublinear verification. In *Proc. ITCS* (2014), ACM, pp. 523–540.

[94] Guo, S., Hubcek, P., Rosen, A., and Vald, M. Rational sumchecks. In *Theory of Cryptography*. Springer, 2016, pp. 319–351.

[95] Gur, T., and Raz, R. Arthur–Merlin streaming complexity. *Information and Computation* (2014).

[96] Gur, T., and Rothblum, R. D. Non-interactive proofs of proximity. In *Proc. ITCS* (2015), T. Roughgarden, Ed., ACM, pp. 133–142.

[97] Har-Peled, S. No coreset, no cry. In *Proc. FSTTCS* (2005), pp. 324–335.

[98] Har-Peled, S. *Geometric approximation algorithms*. American Mathematical Society, 2011.

[99] Har-Peled, S., and Mazumdar, S. On coresets for k-means and k-median clustering. In *Proceedings of the Thirty-sixth Annual ACM Symposium on Theory of Computing* (2004), ACM, pp. 291–300.

[100] Held, M., and Mitchell, J. S. B. Triangulating input-constrained planar point sets. *Information Processing Letters 109*, 1 (2008).

[101] Jayram, T., Kale, S., and Vee, E. Efficient aggregation algorithms for probabilistic data. In *SODA* (2007).

[102] Jayram, T., McGregor, A., Muthukrishnan, S., and Vee, E. Estimating statistical aggregates on probabilistic data streams. In *PODS* (2007).

[103] Jrgensen, A. G., Lffler, M., and Phillips, J. M. Geometric computation on indecisive and uncertain points. arXiv:1205.0273.

[104] Jrgensen, A. G., Lffler, M., and Phillips, J. M. Geometric computation on indecisive points. In *WADS* (2011).

[105] Jowhari, H., and Ghodsi, M. New streaming algorithms for counting triangles in graphs. In *Computing and Combinatorics*. Springer, 2005, pp. 710–716.

[106] Jowhari, H., Salam, M., and Tardos, G. Tight bounds for Lp samplers, finding duplicates in streams, and related problems. In *Proceedings of the Thirtieth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems* (2011), ACM, pp. 49–58.

[107] Kalai, Y. T., Raz, R., and Rothblum, R. D. How to delegate computations: The power of no-signaling proofs. Cryptology ePrint Archive, Report 2013/862, 2013. `http://eprint.iacr.org/`.

[108] Kamousi, P., Chan, T. M., and Suri, S. The stochastic closest pair problem and nearest neighbor search. In *WADS* (2011).

[109] Kamousi, P., Chan, T. M., and Suri, S. Stochastic minimum spanning trees in Euclidean spaces. In *SOCG* (2011).

[110] Kaplan, H., Rubin, N., Sharir, M., and Verbin, E. Counting colors in boxes. In *SODA* (2007).

[111] Kapralov, M. Better bounds for matchings in the streaming model. In *SODA* (2013), SIAM, pp. 1679–1697.

[112] Kapralov, M., Khanna, S., and Sudan, M. Streaming lower bounds for approximating max-cut. In *SODA* (2015), SIAM, pp. 1263–1282.

[113] Kapralov, M., Khanna, S., Sudan, M., and Velingker, A. (1+ ω (1))-approximation to max-cut requires linear space. In *Proceedings of the Twenty-eighth Annual ACM-SIAM Symposium on Discrete Algorithms* (2017), SIAM, pp. 1703–1722.

[114] Kapralov, M., Lee, Y. T., Musco, C., Musco, C., and Sidford, A. Single pass spectral sparsification in dynamic streams. In *Foundations of Computer Science (FOCS), 2014 IEEE 55th Annual Symposium on* (2014), IEEE, pp. 561–570.

[115] Karp, R. M., Upfal, E., and Wigderson, A. Constructing a perfect matching is in random NC. *Combinatorica 6*, 1 (Jan. 1986), 35–48.

[116] Kelner, J. A., and Levin, A. Spectral sparsification in the semi-streaming setting. *Theory of Computing Systems 53*, 2 (2013), 243–262.

[117] Klauck, H. On Arthur Merlin games in communication complexity. In *Computational Complexity (CCC), 2011 IEEE 26th Annual Conference on* (2011), IEEE, pp. 189–199.

[118] Klauck, H. On Arthur Merlin games in communication complexity. In *Proc. CCC* (2011), pp. 189–199.

[119] Klauck, H., and Prakash, V. An improved interactive streaming algorithm for the distinct elements problem. In *Automata, Languages, and Programming*. Springer, 2014, pp. 919–930.

[120] Klauck, H., and Prakash, V. An improved interactive streaming algorithm for the distinct elements problem. In *Proc. ICALP* (2014), pp. 919–930.

[121] Kogan, D., and Krauthgamer, R. Sketching cuts in graphs and hypergraphs. In *Proc. ITCS* (2015), ACM, pp. 367–376.

[122] Knig, D. Gráfok és Alkalmazásuk a Determinánsok és a Halmazok Elméletére. *Matematikai és TermÃľszettudományi Értesít***o** *34* (1916), 104–119.

[123] Konrad, C. Maximum matching in turnstile streams. *arXiv preprint arXiv:1505.01460* (2015).

[124] Kruger, H. Basic Measures for Imprecise Point Sets in $\mathbb{R}^d$. Master's thesis, Utrecht University, 2008.

[125] Li, Y., Long, P. M., and Srinivasan, A. Improved bounds on the samples complexity of learning. *Journal of Computer and System Science 62* (2001), 516–527.

[126] Lffler, M., and Phillips, J. Shape fitting on point sets with probability distributions. In *ESA* (2009), Springer Berlin / Heidelberg.

[127] Lffler, M., and Snoeyink, J. Delaunay triangulations of imprecise points in linear time after preprocessing. In *SOCG* (2008).

[128] Lund, C., Fortnow, L., Karloff, H. J., and Nisan, N. Algebraic methods for interactive proof systems. *J. ACM 39*, 4 (1992), 859–868.

[129] Mathieu, C., and Schudy, W. Yet another algorithm for dense max cut: Go greedy. In *Proceedings of the Nineteenth Annual ACM-SIAM Symposium on Discrete Algorithms* (2008), Society for Industrial and Applied Mathematics, pp. 176–182.

[130] Matousek, J. Approximations and optimal geometric divide-and-conquer. *Journal of Computer and System Sciences 50*, 2 (1995), 203 – 208.

[131] Matousek, J. *Geometric discrepancy*. Springer, 1999.

[132] McGregor, A. Graph stream algorithms: A survey. *ACM SIGMOD Record 43*, 1 (2014), 9–20.

[133] Monemizadeh, M., and Woodruff, D. P. 1-pass relative-error Lp-sampling with applications. In *Proceedings of the Twenty-first Annual ACM-SIAM Symposium on Discrete Algorithms* (2010), SIAM, pp. 1143–1160.

[134] Munteanu, A., Sohler, C., and Feldman, D. Smallest enclosing ball for probabilistic data. In *Proceedings of the Thirtieth Annual Symposium on Computational Geometry* (2014), ACM, p. 214.

[135] Nagai, T., and Tokura, N. Tight error bounds of geometric problems on convex objects with imprecise coordinates. In *Jap. Conf. on Discrete and Comput. Geom.* (2000), LNCS 2098, pp. 252–263.

[136] O'Donnell, R. Almost orthogonal vectors. `http://mathoverflow.net/questions/24864/almost-orthogonal-vectors/24886\#24886`.

[137] Ostrovsky-Berman, Y., and Joskowicz, L. Uncertainty envelopes. In *21st European Workshop on Comput. Geom.* (2005), pp. 175–178.

[138] Pavan, A., Tangwongsan, K., Tirthapura, S., and Wu, K.-L. Counting and sampling triangles from a graph stream. *Proceedings of the VLDB Endowment 6*, 14 (2013), 1870–1881.

[139] Phillips, J. M. Algorithms for ε-approximations of terrains. In *ICALP* (2008).

[140] Phillips, J. M. *Small and stable descriptors of distributions for geometric statistical problems*. PhD thesis, Duke University, 2009.

[141] Razborov, A. A. On the distributional complexity of disjointness. *Theoretical Computer Science 106*, 2 (1992), 385–390.

[142] Reingold, O., Rothblum, G. N., and Rothblum, R. D. Constant-round interactive proofs for delegating computation. In *Proc. STOC* (2016).

[143] Rudelson, M., and Vershynin, R. Sampling from large matrices: An approach through geometric functional analysis. *J. ACM 54*, 4 (July 2007).

[144] Sarma, A. D., Benjelloun, O., Halevy, A., Nabar, S., and Widom, J. Representing uncertain data: Models, properties, and algorithms. *The VLDB Journal 18*, 5 (2009), 989–1019.

[145] Sohler, C. Dortmund workshop on streaming algorithms, problem 52. `http://sublinear.info/index.php?title=Open_Problems:52`, 2012.

[146] Spencer, J., Srinivasan, A., and Tetai, P. Discrepancy of permutation families. Unpublished manuscript, 2001.

[147] Tao, Y., Cheng, R., Xiao, X., Ngai, W. K., Kao, B., and Prabhakar, S. Indexing multi-dimensional uncertain data with arbitrary probability density functions. In *VLDB* (2005).

[148] Thaler, J. Private communication, 2015.

[149] Thaler, J. Semi-streaming algorithms for annotated graph streams. In *Proc. ICALP* (2016).

[150] Tutte, W. T. The factorization of linear graphs. *The Journal of the London Mathematical Society, Ser. 1 22*, 2 (1947), 107–111.

[151] van der Merwe, R., Doucet, A., de Freitas, N., and Wan, E. The unscented particle filter. In *NIPS* (2000), vol. 8, pp. 351–357.

[152] van Kreveld, M., and Lffler, M. Largest bounding box, smallest diameter, and related problems on imprecise points. *Computational Geometry: Theory and Applications 43* (2010), 419–433.

[153] Vapnik, V., and Chervonenkis, A. On the uniform convergence of relative frequencies of events to their ppobabilities. *Theory of Probability and its Applications 16* (1971), 264–280.

[154] Verbin, E., and Yu, W. The streaming complexity of cycle counting, sorting by reversals, and other problems. In *SODA* (2011), SIAM, pp. 11–25.

[155] Yang, S., Zhang, W., Zhang, Y., and Lin, X. Probabilistic threshold range aggregate query processing over uncertain data. *Advances in Data and Web Management* (2009), 51–62.

[156] Zhang, Y., Lin, X., Tao, Y., Zhang, W., and Wang, H. Efficient computation of range aggregates against uncertain location based queries. *IEEE Transactions on Knowledge and Data Engineering 24* (2012), 1244–1258.